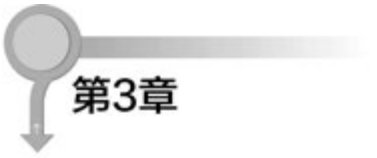


Echarts可视化库

★本章重点★

◎ 了解 Echarts 基本概念和特点
◎ 理解 Echarts 的安装和配置方法
◎ 掌握 Echarts 样式设置
◎ 掌握 Echarts 异步数据加载和更新
◎ 掌握 Echarts 数据集的使用


视频讲解

3.1 Echarts 基础

Echarts 是一个使用 JavaScript 实现的开源可视化库，涵盖各行业图表，满足各种需求。Echarts 遵循 Apache-2.0 开源协议，免费商用。Echarts 兼容当前绝大部分浏览器及多种设备，可随时随地任意展示。学习 Echarts 数据可视化之前需要 HTML 和 JavaScript 的基础知识。

3.1.1 Echarts 概览

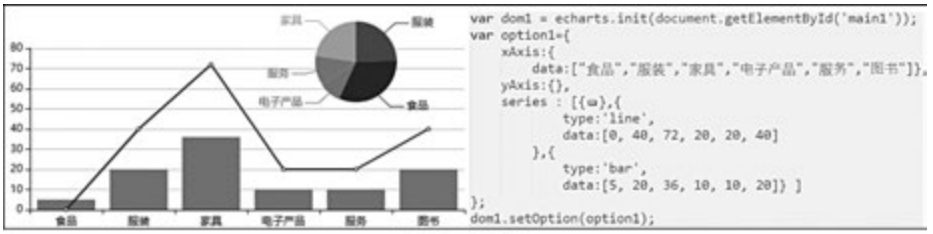
1. Echarts 实例

一个网页中可以创建多个 Echarts 实例。每个 Echarts 实例中可以创建多个图表和坐标系(用 option 来描述)。准备一个 DOM 节点(作为 Echarts 的渲染容器)，就可以在上面创建一个 Echarts 实例。每个 Echarts 实例独占一个 DOM 节点。图 3-1 所示实例有两个 DOM 节点，图 3-1(a)为 dom1 节点，图 3-1(b)为 dom2 节点。

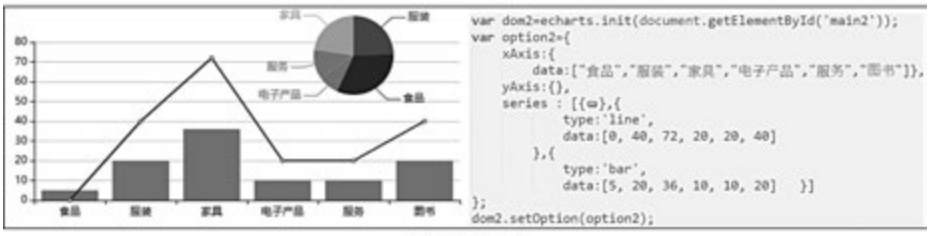
2. 系列(series)

系列(series)是很常见的名词。在 Echarts 里，系列是指一组数值及它们映射成的图。“系列”这个词原本可能来源于“一系列的数据”，而在 Echarts 中取其扩展的概念，不仅表示数据，也表示数据映射成为的图。所以，一个系列包含的要素至少有一组数值、图表类型(series.type)，以及其他关于这些数据如何映射成图的参数。

Echarts 里系列类型(series.type)就是图表类型。系列类型(series.type)至少有 line(折线图)、bar(柱形图)、pie(饼图)、scatter(散点图)、graph(关系图)、tree(树图)等，如图 3-2 所



(a) dom1节点



(b) dom2节点

图 3-1 多 Echarts 实例

示,右侧的 option 中声明了三个系列(series):pie(饼图系列)、line(折线图系列)和 bar(柱形图系列),每个系列中都有它所需要的数据(series.data)。

图 3-2 给出了 3 种图表:饼图、折线图和柱形图,以及每种图表的代码和图表对应关系。

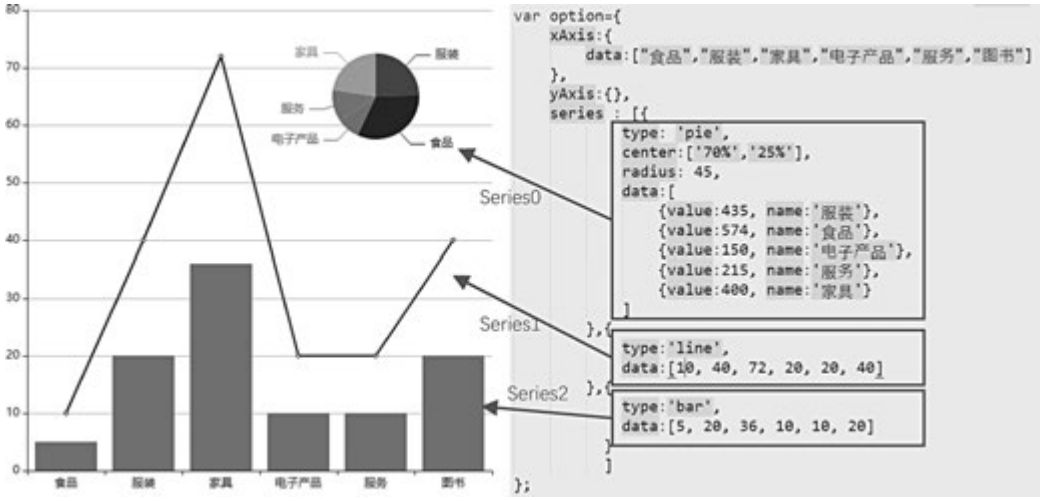


图 3-2 混合图表

3. 组件(component)

在系列之上,Echarts 中各种内容被抽象为“组件”。例如,Echarts 中至少有这些组件:xAxis(直角坐标系 X 轴)、yAxis(直角坐标系 Y 轴)、grid(直角坐标系底板)、angleAxis(极坐标系角度轴)、radiusAxis(极坐标系半径轴)、polar(极坐标系底板)、geo(地理坐标系)、dataZoom(数据区缩放组件)、visualMap(视觉映射组件)、tooltip(提示框组件)、toolbox(工具栏组件)、series(系列)等。其实系列(series)也是一种组件,可以理解为:系列是专门绘制“图”的组件。如图 3-3 所示,右侧的 option 中声明了各个组件(包括系列),各个组件就出现在了图中。

4. 用 option 描述图表

上面已经出现了 option 的概念。Echarts 的使用者,使用 option 来描述其对图表的各种

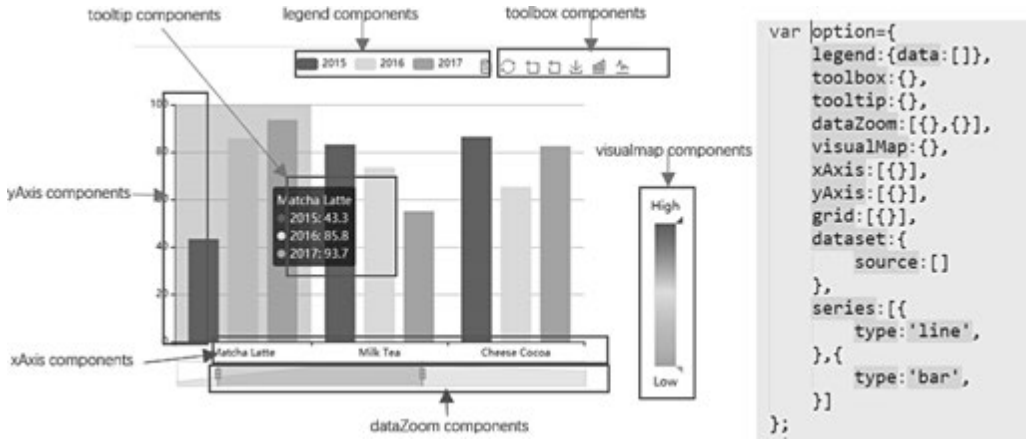


图 3-3 图表组件

需求，包括：有什么数据、要画什么图表、图表长什么样子、含有什么组件、组件能操作什么事情等。简而言之，option 表述了数据、数据如何映射成图形、交互行为。代码和注释如下所示：

```
var dom = document.getElementById('dom-id');      //创建 Echarts 实例
var chart = Echarts.init(dom);
//用 option 是一个大的 JavaScript 对象，描述数据、数据映射成图形、交互行为等
var option = {//option 每个属性是一类组件
  legend: {...},grid: {...},tooltip: {...},toolbox: {...},
  dataZoom: {...},visualMap: {...},
  //如果有多个同类组件，那么就是个数组.例如这里有三个 X 轴。
  xAxis: [ //数组每项表示一个组件实例，用 type 描述“子类型”
      {type: 'category', ...},{type: 'category', ...},{type: 'value', ...}],
  yAxis: [{...}, {...}],
  series: [ //数组构成了多个系列;每个系列使用 type 描述“子类型”，即“图表类型”
      {type: 'line', data: [['AA', 332], ['CC', 124], ['FF', 412], ... ]},
      {type: 'line', data: [2231, 1234, 552, ... ]},
      {type: 'line', data: [[4, 51], [8, 12], ... ]}
  ]};
chart.setOption(option);   //调用 setOption 将 option 输入 Echarts，然后 Echarts 渲染图表
```

系列里的 series.data 是本系列的数据。而另一种描述方式，系列数据从 dataset 中取，如下所示：

```
var option = {
  dataset: {
    source: [[121, 'XX', 442, 43.11],[663, 'ZZ', 311, 91.14],
             [913, 'ZZ', 312, 92.12], ... ]},
  xAxis: {}, yAxis: {},
  series: [
    //数据从 dataset 中取，encode 中的数值是 dataset.source 的维度 index (即第几列)
    {type: 'bar', encode: {x: 1, y: 0}},{type: 'bar', encode: {x: 1, y: 2}},
    {type: 'scatter', encode: {x: 1, y: 3}}, ...]};
```

5. 组件的定位

不同的组件、系列，常有不同的定位方式。

1) 类 CSS 的绝对定位

多数组件和系列都能够基于 top/right/down/left/width/height 绝对定位。这种绝对定位的方式，类似于 CSS 的绝对定位(position：absolute)。绝对定位基于的是 Echarts 容器的 DOM 节点。其中，它们每个值都可以是以下两种取值。

(1) 绝对数值(如 bottom：54 表示：距离 Echarts 容器底边界 54 像素)。

(2) 或者基于 Echarts 容器宽高的百分比(如 right：30%'表示：距离 Echarts 容器右边界的距离是 Echarts 容器宽度的 30%)。

图 3-4 所示是对 grid 组件(也就是直角坐标系的底板)设置 left、right、height、bottom 达到的效果。

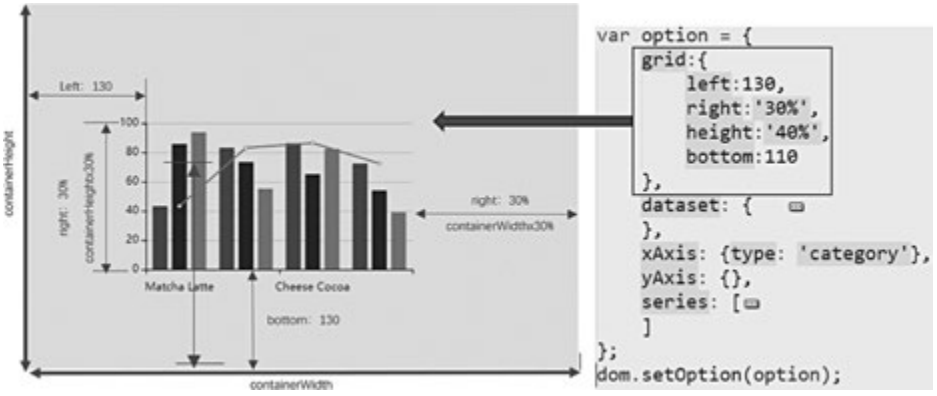


图 3-4　类 CSS 的绝对定位

通过图 3-4 的布局和代码可以看到，left、right、width 是一组(横向)，top、bottom、height 是另一组(纵向)，这两组没有什么关联。每组中至多设置两项就可以了，第三项会被自动算出。例如，设置 left 和 right 就可以了，width 会被自动算出。

2) 中心半径定位

少数圆形的组件或系列可以使用“中心半径定位”，如 pie(饼图)、sunburst(旭日图)、polar(极坐标系)。中心半径定位往往依据 center(中心)、radius(半径)来决定位置。

3) 其他定位

少数组件和系列可能有自己特殊的定位方式，在它们的文档中会有说明。

6. 坐标系

很多系列，如 line(折线图)、bar(柱形图)、scatter(散点图)、heatmap(热力图)等，需要运行在“坐标系”上。坐标系用于布局这些图及显示数据的刻度等。例如，Echarts 中至少支持这些坐标系：直角坐标系、极坐标系、地理坐标系(GEO)、单轴坐标系、日历坐标系等。其他一些系列，如 pie(饼图)、tree(树图)等，并不依赖坐标系，能独立存在。还有一些图，如 graph(关系图)等，既能独立存在，也能布局在坐标系中，依据用户的设定而来。

一个坐标系可能由多个组件协作而成。以最常见的直角坐标系为例，直角坐标系包括 xAxis(直角坐标系 X 轴)、yAxis(直角坐标系 Y 轴)、grid(直角坐标系底板)三种组件。xAxis、yAxis 被 grid 自动引用并组织起来，共同工作。

(1) 最简单的坐标系。图 3-5 所示是最简单的使用直角坐标系的方式：只声明了 xAxis、yAxis 和一个 scatter(散点图系列)，Echarts 就可以自动为它们创建 grid 并关联起它们。

(2) 两个 yAxis，共享了一个 xAxis。两个 series，也共享了这个 xAxis，但是分别使用不同的 yAxis，使用 yAxisIndex 来指定它自己使用的是哪个 yAxis，如图 3-6 所示。

(3) 一个 Echarts 实例包含多个 grid。每个 grid 分别有 xAxis、yAxis，它们使用 xAxisIndex、yAxisIndex、gridIndex 来指定引用关系，如图 3-7 所示。

另外，一个系列往往能运行在不同的坐标系中。例如，一个 scatter(散点图)能运行在直角坐标系、极坐标系、地理坐标系等各种坐标系中。同样，一个坐标系，也能承载不同的系列，

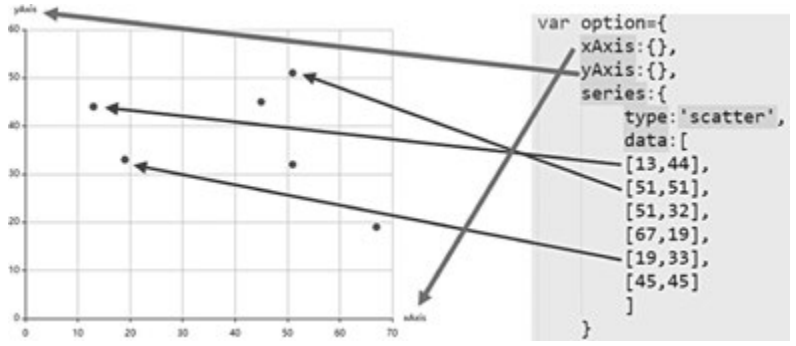


图 3-5 散点图点与坐标系对应关系

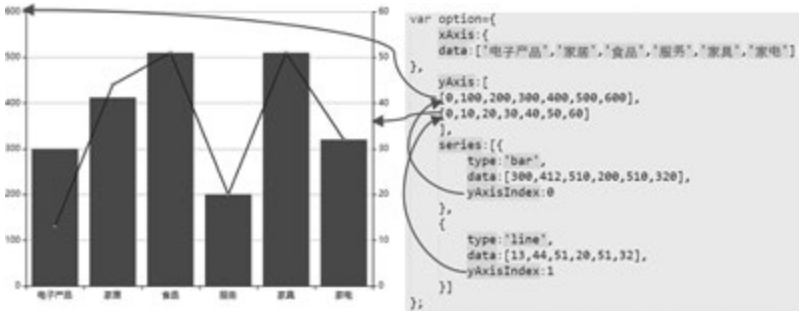


图 3-6 两个 yAxis,共享了一个 xAxis

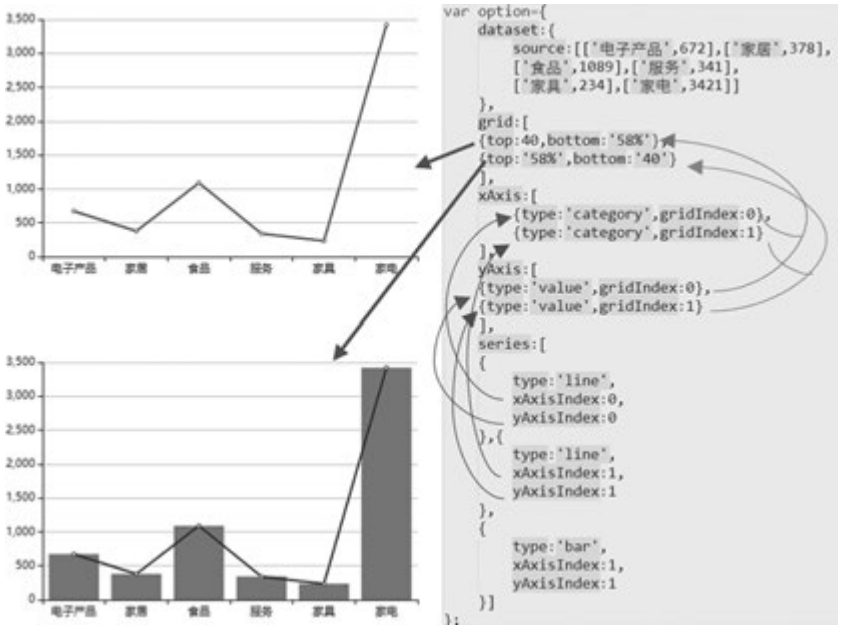


图 3-7 一个 Echarts 实例包含多个 grid

如上面出现的各种例子,直角坐标系里承载了 line(折线图)、bar(柱形图)等。

3.1.2 Echarts 安装与配置

1. 安装

(1) 独立版本。可以直接下载 Echarts. min. js 并用< script >标签引入。另外,开发环境

下可以使用源代码版本 Echarts.js,并用< script >标签引入,源码版本包含了常见的错误提示和警告。

也可以在 Echarts 的官网上直接下载更多丰富的版本,包含了不同主题与语言,主要有完全版(Echarts/dist/Echarts.js)、常用版(Echarts/dist/Echarts.common.js)和精简版(Echarts/dist/Echarts.simple.js)。

(2) 使用 CDN 方法。以下推荐国外比较稳定的两个 CDN,国内还没发现哪一家比较好,可以使用三种在线 js 文件的任何一种,但是目前还是建议下载到本地。

2. 配置

本节的主要目的是为大家介绍使用 Echarts 生成图表的一些配置方法和配置步骤。下面以绘制柱形图为例来展示 Echarts 图表绘制过程。

(1) 创建 HTML 页面:创建一个 HTML 页面,引入 Echarts.min.js。

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <!-- 引入 Echarts 文件 -->
  <script src="Echarts.min.js"></script>
</head>
</html>
```

(2) 为 Echarts 准备具备宽高的 DOM 容器:实例中 id="main"的 div 用于包含 Echarts 绘制的图表。

```
<body>
  <!-- 为 Echarts 准备一个具备大小(宽高)的 DOM -->
  <div id="main" style="width: 600px;height:400px;"></div>
</body>
```

(3) 设置配置信息:Echarts 库使用 JSON 格式来配置,主要包括标题、提示信息、图例组件、x 轴、y 轴、系列列表。

```
title: { text: '第一个 Echarts 实例'} <!-- 为图表配置标题 -->
tooltip: {}, <!-- 为图表配置提示信息 -->
legend: {     //为图表配置图例组件,展现了不同系列的标记(symbol)、颜色和名字
  data: [{
    name: '系列 1',
    icon: 'circle',                  //强制设置图形为圆
    textStyle: {color: 'red'         //设置文本为红色}
  }]
}
xAxis: { data: ["衬衫","羊毛衫","雪纺衫","裤子","高跟鞋","袜子"]}      //x 轴
yAxis: {}                            //y 轴
series: [{                           //系列列表:每个系列通过 type 决定自己的图表类型
  name: '销量',                      //系列名称
  type: 'bar',                       //系列图表类型
  data: [5, 20, 36, 10, 10, 20] }]   //系列中的数据内容
```

【示例 3-1】 绘制电商平台商品销售量的柱形图。

(1) 示例代码如下:

```
<!DOCTYPE html>
<html>
```

```
<head>
    <meta charset="utf-8">
    <title>系列类型</title>
    <script src="Echarts.min.js"></script><!-- 引入 Echarts 文件 -->
</head>
<body>
<!-- 为 Echarts 准备一个具备大小(宽高)的 DOM -->
<div id="main" style="width: 600px;height:600px;"></div>
    <script type="text/javascript">
        var myChart = Echarts.init(document.getElementById('main'));
        var option = {
    title:{text:"柱形图"}
            xAxis:{data:["食品","服装","家具","电子产品","服务","图书"]},
            yAxis:{},
            series : [{
          name:'销量',
          type:'bar',
          data:[5, 20, 36, 10, 10, 20]
          }
          ]};
    myChart.setOption(option);
    </script>
</body>
</html>
```

(2) 保存该 HTML 文件,并在浏览器打开该网页文件,运行结果如图 3-8 所示。

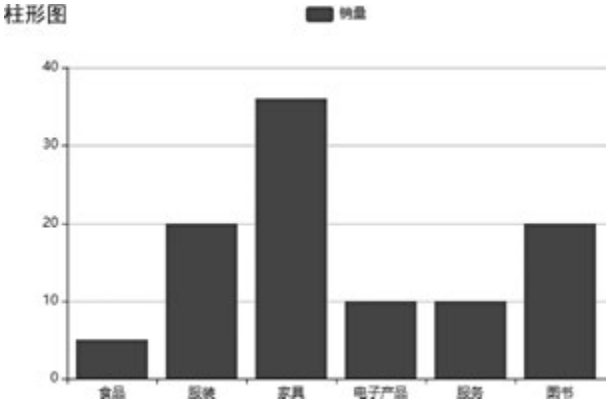


图 3-8 示例 3-1 运行结果

3.2 Echarts 饼图

Echarts 使用 javaScript 实现的开源可视化库,可以流畅地运行在 PC 端和移动设备上,兼容当前绝大部分浏览器(Microsoft Edge、Chrome、Firefox、Safari 等),底层依赖轻量级的矢量图形库,提供直观、交互丰富、可高度个性化定制的数据可视化图表。type 值和对应的图表类型如表 3-1 所示,且可以支持各类图表的混搭。

表 3-1 type 值和对应的图表类型

type 值	图 表 类 型	type 值	图 表 类 型
type: 'bar'	柱形图/条形图	type: 'line'	折线/面积图
type: 'pie'	饼图	type: 'scatter'	散点图

续表

type 值	图 表 类 型	type 值	图 表 类 型
type：'effectScatter'	涟漪特效动画的散点图	type：'radar'	雷达图
type：'tree'	树图	type：'treemap'	树图
type：'sunburst'	旭日图	type：'boxplot'	箱形图
type：'candlestick'	K 线图	type：'heatmap'	热力图
type：'map'	地图	type：'parallel'	平行坐标系的系列
type：'lines'	线图	type：'graph'	关系图
type：'sankey'	桑基图	type：'funnel'	漏斗图
type：'gauge'	仪表盘	type：'pictorialBar'	象形柱图
type：'themeRiver'	主题河流	type：'custom'	自定义系列

3.2.1 饼图

通过前面的 Echarts 相关配置，我们已经学会使用 Echarts 绘制一个简单的柱形图，本节我们将绘制饼图和在饼图基础上的个性化图表样式。饼图主要是通过扇形的弧度表现不同类目的数据在总和中的占比，它的数据格式比柱形图更简单，只有一维的数值，不需要给类目。因为不在直角坐标系中，所以也不需要 xAxis、yAxis。绘制饼图大致有以下 4 个步骤。

步骤 1：在页面先引入 Echarts.js 或 Echarts.min.js，还可以引入 jquery.js，这里面包含了所有的组件。语句为

```
<script src = "Echarts.min.js"></script>
```

步骤 2：在 body 里面建立一个饼图的容器 div，必须有宽高属性。语句为

```
<div id = "dom" style = "width: 400px;height:400px; "></div>
```

步骤 3：对容器进行初始化，即获取 div，用 Echarts.js 进行初始化，语句为

```
var myPie = Echarts.init(document.getElementById("dom"));
```

步骤 4：初始化之后就可以进行参数设置了，语句为

```
var option = {
        title: { text: '电商平台商品饼图'},
        series : [{
              type: 'pie',        //设置图表类型为饼图
            //饼图的半径,外半径为可视区尺寸(容器宽高中较小项)的 55 % 长度
              radius: '55 % ',
              data:[              //数据数组,name 为数据项名称,value 为数据项值
                {value:435, name:'服装'},{value:574, name:'食品'},
                {value:150, name:'电子产品'},{value:215, name:'服务'},
                {value:400, name:'家具'}]
        }]
};
```

步骤 5：使用刚指定的配置项和数据显示图表。语句为

```
myPie.setOption(option);
```

3.2.2 南丁格尔图

Apache Echarts™ 提供了丰富的自定义配置选项，并且能够从全局、系列、数据三个层级去设置数据图形的样式。Echarts 中的饼图支持通过设置 roseType 显示成南丁格尔图，下面我们来看如何使用 Echarts 实现南丁格尔图，南丁格尔图会通过半径表示数据的大小。语句如下，绘制的南丁格尔图如图 3-9 所示。

```
option = {
  series : [
      {
        name: '访问来源',
        type: 'pie',
        radius: '55%',
        roseType: 'angle',
        data:[ … ] }
  ]};
```

1. 阴影的配置

Echarts 中有一些通用的样式，诸如阴影、透明度、颜色、边框颜色、边框宽度等，这些样式一般都会在系列的 itemStyle 里设置。例如，阴影的样式可以通过下面几个配置项设置，阴影效果如图 3-10 所示。

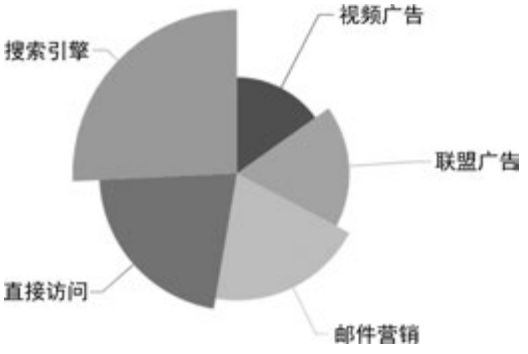


图 3-9 南丁格尔图

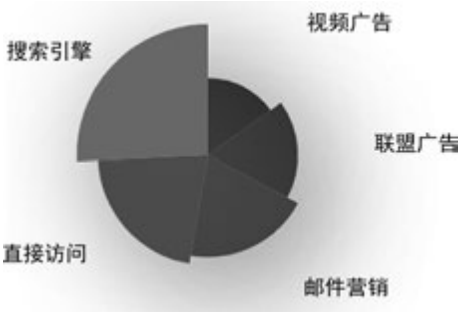


图 3-10 阴影效果

```
itemStyle: {
  shadowBlur: 200,                    //阴影的大小
  //阴影水平方向上的偏移
  shadowOffsetX: 0,
  shadowOffsetY: 0,                   //阴影垂直方向上的偏移
  shadowColor: 'rgba(0, 0, 0, 0.5)'}  //阴影颜色
```

2. 深色背景和浅色标签

若想让整个主题改成深色主题，就需要改背景色和文本颜色。深色背景图如图 3-11 所示。

(1) 背景色是全局的，所以直接在 option 下设置 backgroundColor。

```
backgroundColor: '#2c343c'
```

(2) 文本的样式可以设置全局的 textStyle。

```
textStyle: { color: 'rgba(255, 255, 255, 0.3)'}
```

(3) 也可以每个系列分别设置,每个系列的文本设置在 label.textStyle。

```
label: {textStyle: { color: 'rgba(255, 255, 255, 0.3)' }}
```

(4) 饼图还要将标签的视觉引导线的颜色设为浅色。

```
labelLine: {lineStyle: { color: 'rgba(255, 255, 255, 0.3)' }}
```

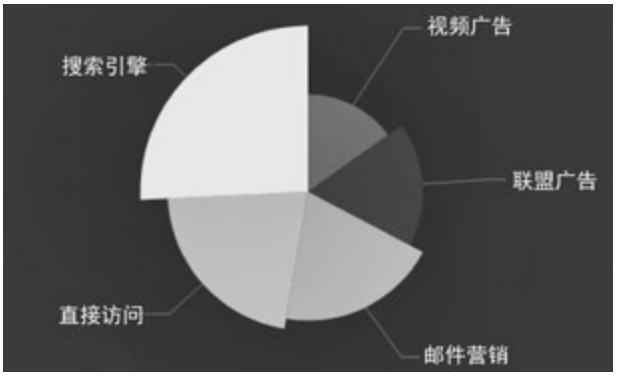


图 3-11 深色背景图

3.3 Echarts 样式设置

Echarts 可以通过样式设置来改变图形元素或者文字的颜色、明暗、大小等。本节主要介绍颜色主题(Theme)、调色盘、直接样式设置(itemStyle、lineStyle、areaStyle、label、…)和视觉映射(visualMap),它们的功能范畴可能会有交叉(即同一种细节的效果可能可以用不同的方式实现),但是它们各有各的场景偏好。

3.3.1 颜色主题

最简单的更改全局样式的方式是直接采用颜色主题(theme)。Echarts4 开始,新内置了两套主题,分别为'light'和'dark',如图 3-12(a)和图 3-12(b)所示。语句如下:

```
var chart = Echarts.init(dom, 'light');或者 var chart = Echarts.init(dom, 'dark');
```

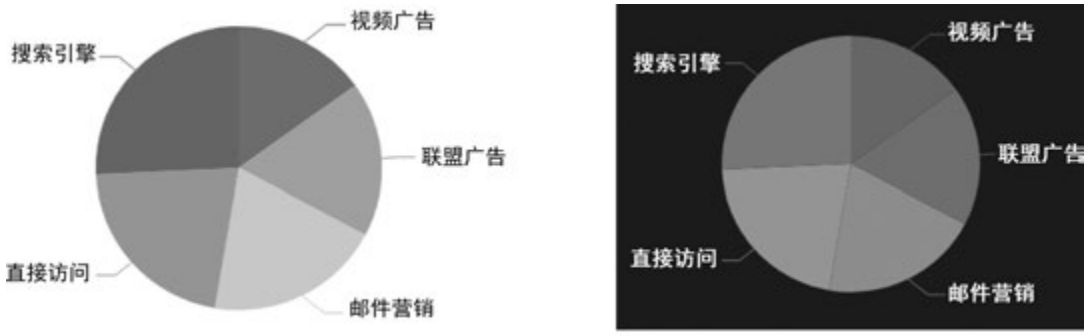


图 3-12 Echarts 内置主题

另外，也可以在 Echarts 官方的主题编辑器选择自己喜欢的主题下载，如图 3-13 所示。下载下来的主题，如果保存为 JSON 文件，则可以自行加载和注册，例如：

```
$.getJSON('xxx/xxx/ vintage.json', function (themeJSON) {
  Echarts.registerTheme('vintage', JSON.parse(themeJSON))
  var chart = Echarts.init(dom, 'vintage');});
```

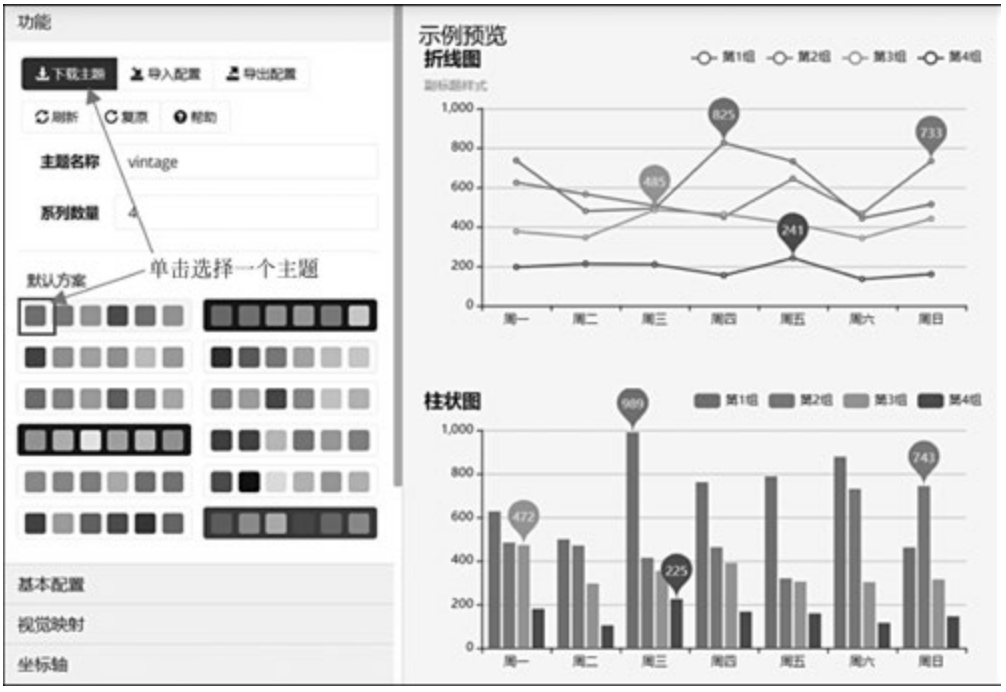


图 3-13 Echarts 官网主题下载

如果保存为 UMD 格式的 JS 文件，应先引入 JS 文件，即引入 vintage.js 文件（假设主题名称是 vintage），然后在 Echarts 实例中使用 vintage 主题，如图 3-14 所示。

```
<script src='Echarts.min.js'></script>
var chart = Echarts.init(dom, 'vintage');
```

图 3-14 JS 文件使用主题

【示例 3-2】 使用 vintage 主题绘制南丁格尔图。

(1) 代码语句如下所示：

```
<!DOCTYPE html>
<html>
  <head>
      <meta charset="utf-8">
      <title>vintage 主题</title>
  <script src="echarts.min.js"></script><!-- 引入 echarts.js -->
  </head>
  <body>
      <!-- 为 ECharts 准备一个具备大小(宽高)的 DOM -->
  <div id="main" style="width: 400px;height:400px;"></div>
      <script type="text/javascript">
            //基于准备好的 DOM,初始化 Echarts 实例
            var myChart = echarts.init(document.getElementById('main'), 'vintage');
            myChart.setOption({
                series: [{
                    name: '访问来源',
                    type: 'pie',//设置图表类型为饼图
                    radius: '55%',//饼图的半径,外半径为可视区尺寸的 55% 长度
                    data: [//数据数组,name 为数据项名称,value 为数据项值
                    {value: 235,name: '视频广告'},{value: 274,name: '联盟广告'},
                    {value: 310,name: '邮件营销'},{value: 335,name: '直接访问'},
                    {value: 400,name: '搜索引擎'}]
                }]
            })
        </script>
    </body>
</html>
```

(2) 运行结果如图 3-15 所示。

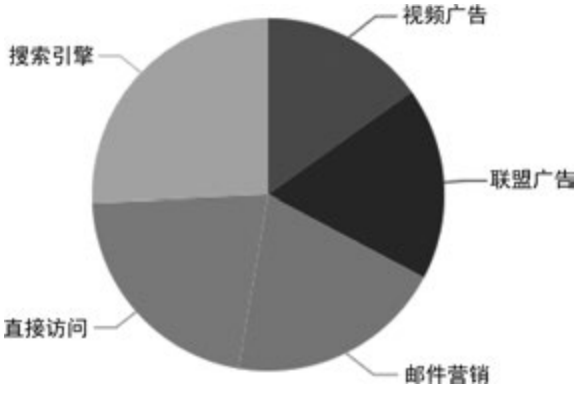


图 3-15 示例 3-2 运行结果

3.3.2 调色盘

调色盘可以在 option 中设置。它给定了一组颜色,图形、系列会自动从其中选择颜色。可以设置全局的调色盘,也可以设置系列自己专属的调色盘。下面通过两个实例来展示全局调色盘和系列专属调色盘的设置。

【示例 3-3】 使用全局调色盘。

(1) 语句代码如下所示：

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>全局调色盘</title>
    <!-- 引入 echarts.js -->
    <script src="echarts.min.js"></script>
</head>
<body>
<div id="main" style="width: 600px;height:600px;"></div>
    <script type="text/javascript">
        var dom = echarts.init(document.getElementById('main'));
        var option={
        color: ['#e69d87','#8dc1a9','#ea7e53', '#eedd78','#73a373','#73b9bc','#7289ab', '#91ca8c','#f49f42'],
            dataset: {
            source: [
                ['product', '2015', '2016', '2017'],
                ['Matcha Latte', 43.3, 25.8, 43.7],
                ['Milk Tea', 83.1, 73.4, 55.1],
                ['Cheese Cocoa', 36.4, 65.2, 22.5],
                ['Walnut Brownie', 32.4, 23.9, 18.1]
            ]
        },
        xAxis: {type: 'category'},
        yAxis: {},
        series: [
            {type: 'bar'},
            {type:'bar'},
            {type:'bar'},
            {
                    type: 'pie',
                    center:['70%','25%'],
                    radius: 45,
                    data:[
                        {value:435, name:'服装'},
                        {value:574, name:'食品'},
                        {value:150, name:'电子产品'},
                        {value:215, name:'服务'},
                        {value:400, name:'家具'}
                    ]
                }
        ]
    };
        dom.setOption(option);
        </script>
</body>
</html>
```

全局调色盘

（2）使用全局调色盘后的运行结果如图 3-16 所示。

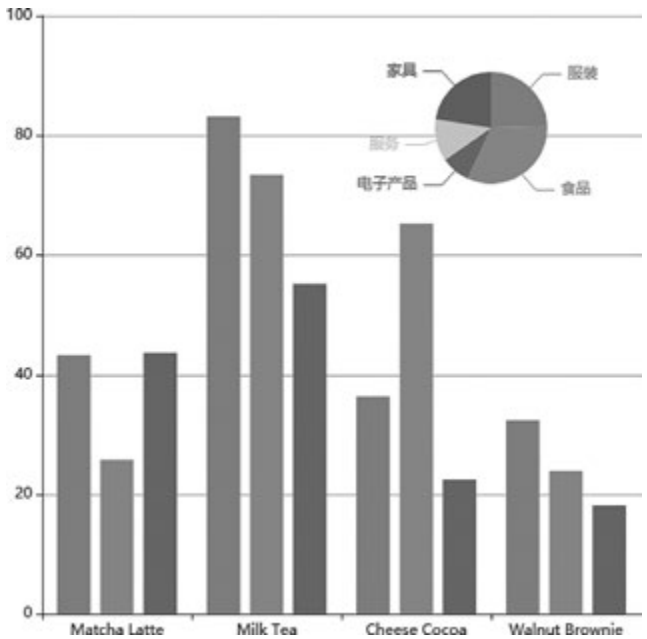


图 3-16　使用全局调色盘后的运行结果

【示例 3-4】 使用系列专属调色盘。

(1) 语句代码如下所示：

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>系列专属调色盘</title>
    <!-- 引入 echarts.js -->
    <script src="echarts.min.js"></script>
</head>
<body>
<div id="main" style="width: 600px;height:600px;"></div>
    <script type="text/javascript">
        var dom = echarts.init(document.getElementById('main'));
       var option={
            dataset: {
            source: [
                ['product', '2015', '2016', '2017'],
                ['Matcha Latte', 43.3, 25.8, 43.7],
                ['Milk Tea', 83.1, 73.4, 55.1],
                ['Cheese Cocoa', 36.4, 65.2, 22.5],
                ['Walnut Brownie', 32.4, 23.9, 18.1]
            ]
        },
        xAxis: {type: 'category'},
        yAxis: {},
        series: [
            {type:'bar',color: '#dd6b66'},
            {type:'bar',color: '#759aa0'},
            {type:'bar',color: '#e69d87'},
            {
                type: 'pie',
                color: ['#fb7293', '#E062AE', '#E690D1', '#e7bcf3', '#9d96f5', '#8378EA', '#968BFFF'],
                center:['70%','25%'],
                radius: 45,
                data:[
                    {value:435, name:'服装'},
                    {value:574, name:'食品'},
                    {value:150, name:'电子产品'},
                    {value:215, name:'服务'},
                    {value:400, name:'家具'}
                ]
            }
        ]
    };
        dom.setOption(option);
        </script>
</body>
</html>
```

柱形图专属调色盘

饼图专属调色盘

(2) 使用系列专属调色盘后的运行结果如图 3-17 所示。

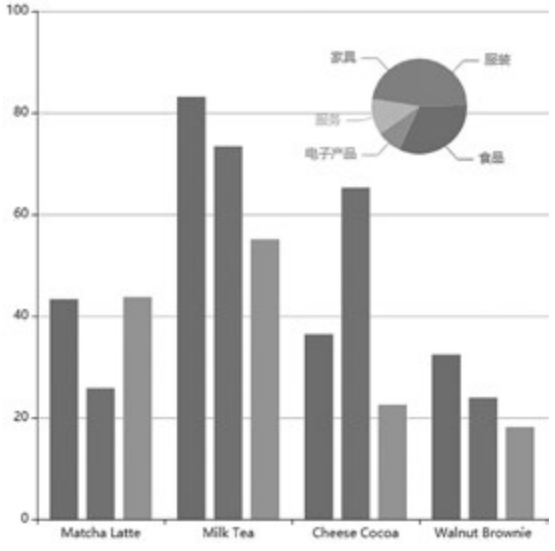


图 3-17 使用系列专属调色盘后的运行结果

3.3.3 直接样式和高亮样式

1. 直接样式设置

直接样式设置是比较常用的设置方式。纵观 Echarts 的 option 中，很多地方都可以设置 areaStyle、lineStyle、label、itemStyle 等属性。这些地方可以直接设置图形元素的颜色、线宽、点的大小、标签的文字、标签的样式等。

(1) areaStyle 属性。可以设置折线下方区域颜色，可以简单地进行纯色填充，设置 color(填充颜色)、opacity(透明度 0～1)和 origin(填充的位置：'start')属性，如图 3-18 所示；也可以设置为过渡颜色，如图 3-19 所示。

```
option = {
xAxis: {
    type: 'category',
    boundaryGap: false,
    data: ['Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun']
},
yAxis: {
    type: 'value'
},
series: [{
    data: [820, 932, 901, 934, 1290, 1330, 1320],
    type: 'line',
    areaStyle: {
                color: 'blue',
                opacity:0.4,
                origin:'start'
                }
            }]
    };
```

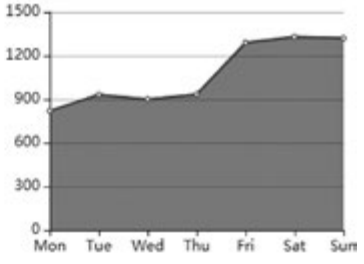


图 3-18 折线下方区域纯色填充

彩色图片

```
areaStyle: {
                color: {
                type: 'linear',
                x0: 0, y0: 0,
                x2: 0, y2: 1,
                colorStops: [
                {offset: 0,color: 'red'},
                {offset: 1,color: 'blue',}
                ],
                globalCoord: false
                        }
                }
        }]
```

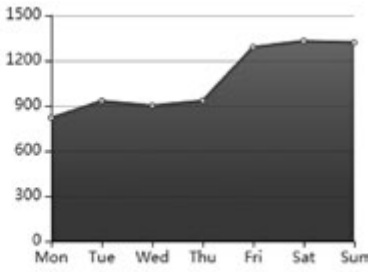


图 3-19 折线下方区域自上而下红色向蓝色过渡填充

(2) lineStyle 属性。可以设置线条的宽度(width)、颜色(color)和线型(type 取值可以为：'solid'：实线；'dotted'：点线；'dashed'：虚线)。

(3) label 属性。可以设置文本的字体(fontFamily)、颜色(color)和大小(fontSize)。

(4) itemStyle 属性。可以设置饼图扇形区域样式，包括 shadowColor、shadowBlur、shadowOffsetX、shadowOffsetY。其中 shadowColor 定义阴影颜色样式，shadowBlur 定义阴影模糊系数(属性默认为 0，没有模糊)，shadowOffsetX 定义阴影 x 轴偏移量，shadowOffsetY 定义阴影 y 轴偏移量。

2. 高亮样式设置

在鼠标悬浮到图形元素上时，一般会出现高亮样式。默认情况下，高亮样式是根据普通样式自动生成的。但是高亮样式也可以自己定义，主要通过 emphasis 属性来定制。emphasis 中的结构和普通样式相同，高亮样式设置方式如图 3-20 所示。

事实上，多数情况下，使用者只会配置普通状态下的样式，而使用默认的高亮样式。所以在 Echarts4 中，支持不写 normal 的配置方法，这样可使配置项更扁平简单。

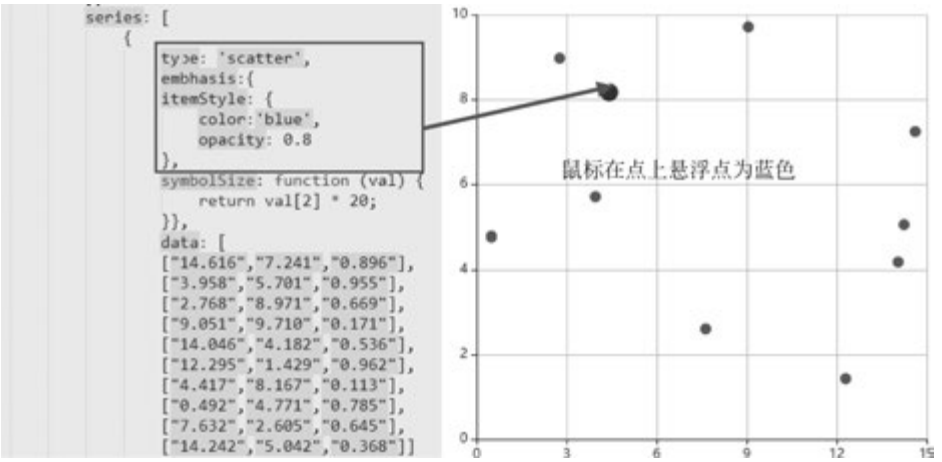


图 3-20　高亮样式设置方式

【示例 3-5】 使用直接和高亮样式的图形。

在柱形图部分通过 lineStyle 属性设置 x 轴和 y 轴的线的颜色和宽度；在饼图部分通过在 label 和 itemStyle 属性中设置 emphasis 部分的相关参数，实现当鼠标悬浮在某个扇区中时，扇区添加边框及扇区文本字体和颜色的变化，运行结果和代码解释如图 3-21 所示。

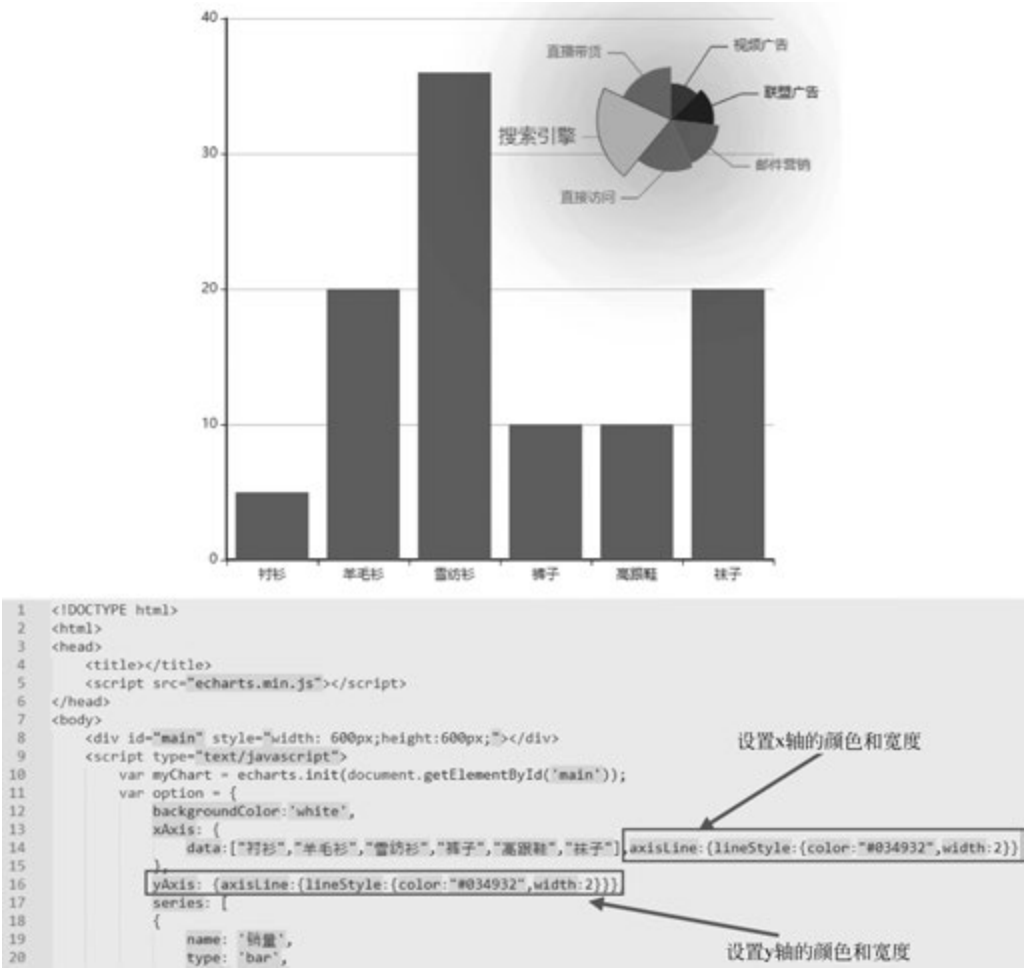


图 3-21　示例 3-5 运行结果和代码解释

图 3-21 （续）

（1）第 14 行代码中的 axisLine:{lineStyle:{color:"#034932",width:2}设置 x 轴的颜色和宽度。

（2）第 15 行代码中的 yAxis:{axisLine:{lineStyle:{color:"#034932",width:2}}},设置 y 轴的颜色和宽度。

（3）第 32～35 行代码如下，设置鼠标在饼图悬浮时的字体、大小和颜色；设置饼图区域线条的颜色和线形。

```
label:{emphasis:
{fontSize:17,fontFamily:"Arial",color:'rgba(65, 59, 117, 0.7)'}},
        itemStyle:{
            emphasis:{borderColor:'rgba(65, 59, 117, 0.7)'},
```

（4）第 36 行代码，normal:{shadowBlur:150,shadowColor: 'black'},设置饼图在正常模式下阴影模糊程度和阴影的颜色。

3.4 Echarts 异步数据加载和更新

前面示例中的数据是初始化后在 setOption 中直接填入的，但是很多时候可能数据需要异步加载后再填入。

3.4.1 异步数据加载

Apache Echarts™ 中实现异步数据的更新非常简单，在图表初始化后不管任何时候只要通过 jQuery 等工具异步获取数据后通过 setOption 填入数据和配置项就行。或者先设置完其他的样式，显示一个空的直角坐标轴，然后获取数据后填入数据。通过如下案例展示异步加载 JSON 数据的方法。

【示例 3-6】 异步加载本地 JSON 文件数据。

（1）JSON 文件。在当前网页文件下一级目录 datas 下创建一个 data1.json 文件，文件内容如下：

```
{
"data": [5, 20, 36, 10, 12, 20],
"category": ["衬衫", "羊毛衫", "雪纺衫", "裤子", "高跟鞋", "袜子"]
}
```

(2) 网页文件代码语句如下：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>json</title>
        <script src="jquery.min.js"></script>
        <script src="Echarts.min.js"></script>
    </head>
    <body>
        <!-- 为Echarts准备一个具备大小(宽高)的DOM -->
        <div id="main" style="width: 600px;height:400px;"></div>
        <script type="text/javascript">
            $(document).ready(function() {
                var myChart = Echarts.init(document.getElementById("main"));
                myChart.setOption({
                    title: {text: '异步加载的数据'},                    //图标题
                    legend: {data: ['销售']},
                    xAxis: {data: [],},                                 //x轴属性
                    yAxis: {},                                          //y轴属性
                    series: [{                                          //图属性
                        name: '销售',
                        type: 'bar',                                    //图类型:柱形图
                        color: ['#C0FF3E'],                             //设置图像颜色
                        data: [],                                       //图表的数值
                    }]
                });
            //使用jQuery中的$.get()获取data1.json文件
            //使用done函数;done(function(data))中data表示调用的函数返回的数据
                $.getJSON('datas/data1.json').done(function(data) {
                    myChart.setOption({
                        xAxis: {data: data.category},
                        series: {name: '销售',data: data.data}});
                })
            })
        </script>
    </body>
</html>
```

(3) Echarts 柱形图运行结果如图 3-22 所示。

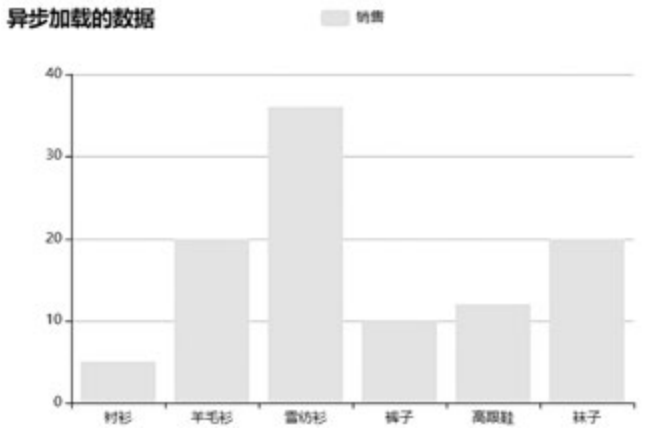


图 3-22　Echarts 柱形图运行结果

(4) 说明：本案例中读取本地 JSON 文件，在浏览器请求中会出现跨域访问资源的问题，如果跨域请求被阻止，有可能使 css、js、ajax 请求、font 字体等资源出现无法正常访问的问题。解决这一问题的方案有很多种，在此，只介绍一种相对简单的方法，以火狐浏览器为例，在地址栏输入“about:config”后，单击“接受风险并继续”按钮，在搜索框中输入 security.fileuri.strict_origin_policy 后，并设置该项为 false，则会载入读取的 JSON 文件数据，正确地显示图形。

3.4.2 loading 动画

如果数据加载时间较长，一个空的坐标轴放在画布上会让用户觉得是不是产生 bug 了，因此需要一个 loading 的动画来提示用户数据正在加载。Echarts 默认提供一个简单的加载动画，只需要调用 showLoading 方法显示即可。数据加载完成后再调用 hideLoading 方法隐藏加载动画。loading 动画效果如图 3-23 所示，语句代码如下：

```
myChart.showLoading();
$.get('datas/data1.json').done(function (data) {
    myChart.hideLoading();
    myChart.setOption(...);
});
```

图 3-23 loading 动画效果

3.4.3 数据的动态更新

Echarts 由数据驱动，数据的改变驱动图表展现的改变，因此动态数据的实现也变得异常简单。所有数据的更新都通过 setOption 实现，只需要定时获取数据，setOption 会填入数据，而不用考虑数据到底产生了哪些变化，Echarts 会找到两组数据之间的差异然后通过合适的动画去表现数据的变化。

通过如下示例展示数据的动态更新效果和数据的设置。

【示例 3-7】 绘制折线图实现数据的动态更新。

(1) 网页文件代码语句如下：

```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <meta charset="utf-8">
```

```
    <title>数据的动态更新</title>
    <script src = "jquery.min.js"></script>
    <script src = "Echarts.min.js"></script>
</head>
<body>
    <div id = "main" style = "width: 600px;height:400px;"></div>
    <script type = "text/javascript">
        var base =  + new Date(2025, 3, 9);     //定义基准日期
        var oneDay = 24 * 60 * 60 * 1000;       //一天的毫秒数
        var date = [];                          //初始化数组,存储日期
        var data = [Math.random() * 150];       //初始指定日期对应的折线图数值
        var now = new Date(base);               //定义当前日期为基准日期
        //实现日期的变化,每次在月份上加 1
        function addData(shift) {
            now = [now.getFullYear(), now.getMonth() + 1,
now.getDate()].join('/');
            date.push(now);                     //将当前日期存入数组 date
            //将当前折线图点值存入数组 data
            data.push((Math.random() - 0.4) * 10 + data[data.length - 1]);
            if (shift) {
//shift() 方法把日期数组的第一个元素从其中删除,并返回第一个元素的值
                date.shift();
//shift() 方法把当前日期对应的 y 数组的第一个元素从其中删除,并返回第一个元素的值
                data.shift();
            }
            //重新获取当前日期,为原来当前日期 + 1 天
            now = new Date( + new Date(now) + oneDay);
        }
        for (var i = 1; i < 50; i++) {
            addData();}
        var option = {
            xAxis: {
                type: 'category',
                //boundaryGap 值为 false 的时候,折线第一个点在 y 轴上
                boundaryGap: false,
                data: date                      //x 轴为日期 },
            yAxis: {boundaryGap: [0, '50 % '],type: 'value'},
            series: [{
                name: '成交',type: 'line',smooth: true,symbol: 'none',
                stack: 'a',
                areaStyle: { normal: {} },
                data: data }],
        };
        setInterval(function() {
            addData(true);
            myChart.setOption({
                xAxis: {data: date},
                series: [{name: '成交',data: data}]
            });
        }, 500);
        var myChart = Echarts.init(document.getElementById('main'));
        myChart.setOption(option)
    </script>
</body>
```

(2) Echarts 动态数据更新的折线图如图 3-24 所示。

最后简单总结示例 3-7 中 Echarts 的折线图表的 xAxis.boundaryGap 或 yAxis.boundaryGap 属性。该属性对于类目轴和非类目轴的设置和表现不一样。boundaryGap 属性如表 3-2 所示。

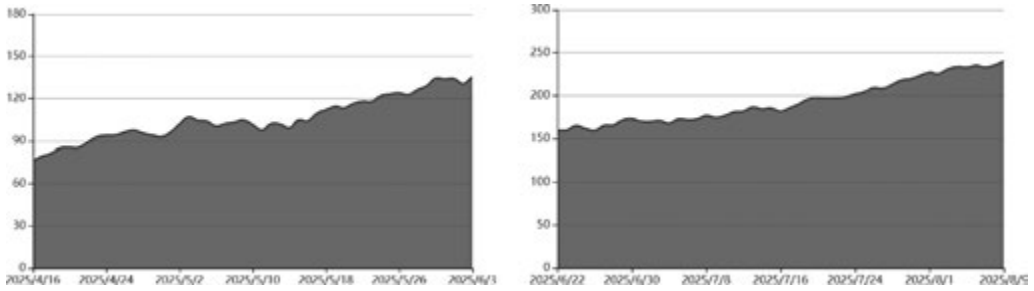


图 3-24 Echarts 动态数据更新的折线图

表 3-2 boundaryGap 属性

分 类	取 值	描 述
类目轴	true(默认值)	x 轴或 y 轴轴两边留白
	false	x 轴或 y 轴轴两边不留白
非类轴 (数值和时间型数值)	[0,0](默认值) [mim,max]	boundaryGap 属性数组内数值采用百分比格式,默认值为数组[0,0]。用户可以根据具体样式来设置,设置的方法为[原始数据最小值与最终最小值的差额,原始数据最大值与最终最大值的差额],所以这个值应该针对绘图的数据范围取值。例如:折线图的 y 轴数据范围为[0,300],那么[0.2,'50%']就表示,在最小值下方扩展 20%空间,在最大值上方扩展范围 50%空间,范围为[0−0.2 * (450−0),450+0.5 * (450−0)],即 y 轴最小坐标为−225,最大坐标为 675

3.5 Echarts 数据集

视频讲解

Apache Echarts™ 4 开始支持 dataset 组件用于单独的数据集声明,从而数据可以单独管理,被多个组件复用,并且可以基于数据指定数据到视觉的映射,Echarts 4 以前,数据只能声明在各个“系列(series)”中。这种方式的优点是,直观易理解,适用于对一些特殊图表类型进行一定的数据类型定制,但缺点是,为匹配这种数据输入形式,常需要数据处理过程把数据分割设置到各个系列和类目轴中,不利于多个系列共享一份数据,也不利于基于原始数据进行图表类型、系列的映射安排。Echarts 4 提供了数据集 dataset 组件来单独声明数据,它带来如下效果。

(1) 能够贴近数据可视化常见思维方式:提供数据,指定数据到视觉的映射,从而形成图表。

(2) 数据和其他配置可以被分离开来,数据常变,其他配置常不变。

(3) 数据可以被多个系列或者组件复用,对于大数据量的场景,不必为每个系列创建一份数据。

(4) 支持更多数据的常用格式,如二维数组、对象数组等,可以避免数据格式转换。

3.5.1 数据集和系列的对应关系

数据集与系列的对应关系分为以下四种情况,分别通过示例分析和讲解。

1. 一行数据对应一个系列

当数据集中只有一行数据,系列中也只有一个系列时,坐标系中什么也没有,默认情况下,类目轴对应 dataset 第一列。注意这里的 x 轴是类目轴,如图 3-25 所示。

2. 一行数据对应多个系列

通过上面的示例可以看到,虽然坐标系中什么也没有,但是上面还是有一个 2023,当添加多个系列后,会根据系列数量添加 legend,如图 3-26(a)所示,但是当系列过多时,只会根据第一行数据来增加上面的 legend,如图 3-26(b)所示。


```
var option = {
    legend: {},//图例组件
    tooplip:{},
    dataset:{  //提供一份数据集
        source: [
        ['product', '2023', '2024', '2025'],
        ]
    },
    xAxis:{type: 'category'},
    yAxis:{},
    series: [//默认情况下，每个系列会自动对应到dataset的每一列
        {type: 'bar'},
    ]
};
```

图 3-25　一行数据对应一个系列


```
var option = {
    legend: {},//图例组件
    tooplip:{},
    dataset:{  //提供一份数据集
        source: [
        ['product', '2023', '2024', '2025'],
        ]
    },
    xAxis:{type: 'category'},
    yAxis:{},
    series: [//默认情况下，每个系列会自动对应到dataset的每一列
        {type: 'bar'},
        {type: 'bar'}
    ]
};
```

(a) 两个数据系列

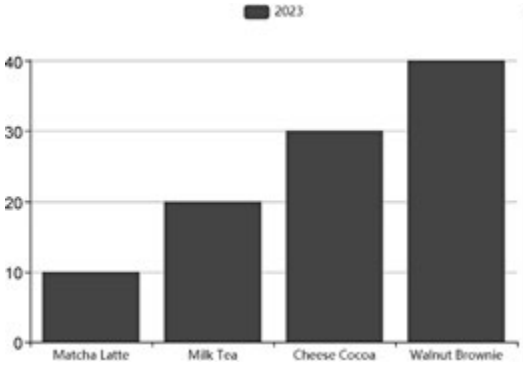

```
var option = {
    legend: {},//图例组件
    tooplip:{},
    dataset:{  //提供一份数据集
        source: [
        ['product', '2023', '2024', '2025'],
        ]
    },
    xAxis:{type: 'category'},
    yAxis:{},
    series: [//默认情况下，每个系列会自动对应到dataset的每一列
        {type: 'bar'},
        {type: 'bar'},
        {type: 'bar'}
    ]
};
```

(b) 三个数据系列

图 3-26　一行数据对应多个系列

3. 多行数据对应一个系列

当数据集中有多行数据，但只有一个系列时，柱状表的个数取决于数据的行数，x 轴的值是数据的第一列元素，y 轴的值是数据的第二列元素值，如图 3-27 所示。

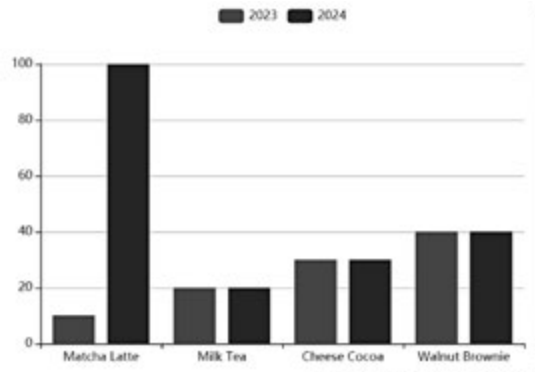



```
var option = {
    legend: {},//图例组件
    tooplip:{},
    dataset:{  //提供一份多行数据集
        source: [
        ['product', '2023', '2024', '2025'],
        ['Matcha Latte', 10, 100, 50],
        ['Milk Tea', 20, 20, 200],
        ['Cheese Cocoa', 30, 30, 60],
        ['Walnut Brownie', 40, 40, 70]
        ]
    },
    xAxis:{type: 'category'},
    yAxis:{},
    //x轴的坐标是数据据的第一列数据，y轴的值是数据的第二列元素值
    series: [
        {type: 'bar'},
    ]
};
```

图 3-27　多行数据对应一个系列

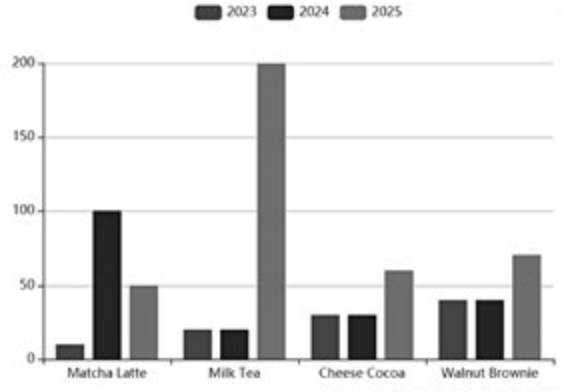
4. 多行数据对应多个系列

通过上面的3种情况，了解了数据集和系列的对应关系。当多行数据对应多个系列时，每一个系列对应的值是数据集从左向右逐列对应的。3列数据对应2个系列如图3-28(a)所示，分别对应第2和第3列数据；3列数据对应3个系列如图3-28(b)所示，分别对应全部的2～4列数据，3列数据对应6个系列如图3-28(c)所示，多出来的系列y轴数据值均为第2列的值。



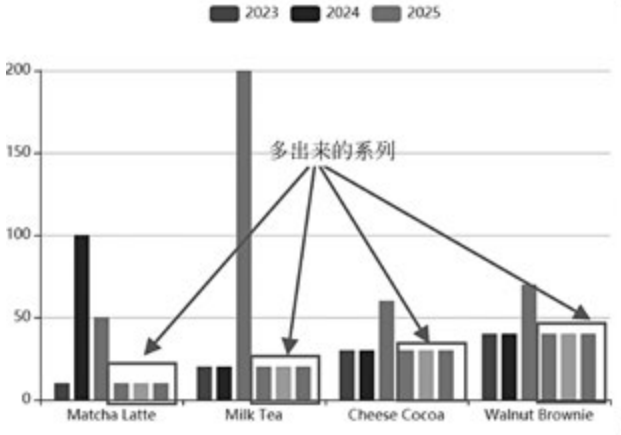

```
var option = {
    legend: {},//图例组件
    tooplip:{},
    dataset:{  //提供一份多行数据集
        source: [
        ['product', '2023', '2024', '2025'],
        ['Matcha Latte', 10, 100, 50],
        ['Milk Tea', 20, 20, 200],
        ['Cheese Cocoa', 30, 30, 60],
        ['Walnut Brownie', 40, 40, 70]
        ]
    },
    xAxis:{type: 'category'},
    yAxis:{},
    series: [
        {type: 'bar'},
        {type: 'bar'}
    ]
};
```

(a) 3列数据对应2个系列




```
var option = {
    legend: {},//图例组件
    tooplip:{},
    dataset:{  //提供一份多行数据集
        source: [
        ['product', '2023', '2024', '2025'],
        ['Matcha Latte', 10, 100, 50],
        ['Milk Tea', 20, 20, 200],
        ['Cheese Cocoa', 30, 30, 60],
        ['Walnut Brownie', 40, 40, 70]
        ]
    },
    xAxis:{type: 'category'},
    yAxis:{},
    series: [
        {type: 'bar'},
        {type: 'bar'},
        {type: 'bar'}
    ]
};
```

(b) 3列数据对应3个系列




```
var option = {
    legend: {},//图例组件
    tooplip:{},
    dataset:{  //提供一份多行数据集
        source: [
        ['product', '2023', '2024', '2025'],
        ['Matcha Latte', 10, 100, 50],
        ['Milk Tea', 20, 20, 200],
        ['Cheese Cocoa', 30, 30, 60],
        ['Walnut Brownie', 40, 40, 70]
        ]
    },
    xAxis:{type: 'category'},
    yAxis:{},
    series: [
        {type: 'bar'},
        {type: 'bar'},
        {type: 'bar'},
        {type: 'bar'},
        {type: 'bar'},
        {type: 'bar'}
    ]
};
```

(c) 3列数据对应6个系列

图3-28 多列数据与多个系列的对应关系

简单概括，数据集中除第一行元素外，每一行数据对应x轴一个类目，每列数据对应一个系列，有多少个系列，一个类目中就有多少图。

3.5.2 维度

常用图表所描述的数据大部分是二维表结构,上述例子中,我们都使用二维数组来容纳二维表。现在,当我们把系列(series)对应到“列”的时候,那么每一列就称为一个维度(dimension),而每一行称为数据项(item)。反之,如果我们把系列(series)对应到表行,那么每一行就是维度(dimension),每一列就是数据项(item)。

维度可以有单独的名字,便于在图表中显示。维度名(dimension name)可以定义在 dataset 的第一行(或者第一列)。例如,图 3-28 中,'Matcha Latte'、'Milk Tea'、'Cheese Cocoa'、'Walnut Brownie'就是维度名。从第二行开始,才是正式的数据。dataset.source 中第一行(列)到底包含不包含维度名,Echarts 默认会自动探测。当然也可以用 dataset.sourceHeader:true 显式声明第一行(列)就是维度,或者 dataset.sourceHeader:false 表明第一行(列)开始直接是数据。

维度也可以使用单独的 dataset.dimensions 或者 series.dimensions 来定义,这样可以同时指定维度名和维度的类型(dimension type)。

大多数情况下,我们并不需要去设置维度类型,因为程序会自动判断。但是如果数据为空而导致判断不够准确时,可以手动设置维度类型。dimension type 取值类型如表 3-3 所示。

表 3-3 dimension type 取值类型

数据类型	描述
'number'	默认,表示普通数据
'ordinal'	对于类目、文本这些 string 类型的数据,如果需要在数轴上使用,则必须是 'ordinal' 类型。Echarts 默认会自动判断这个类型,但是自动判断也可能不完备,所以使用者也可以手动强制指定
'time'	时间数据,能支持自动解析数据成时间戳(timestamp),比如该维度的数据是 '2017-05-10',会自动被解析。如果这个维度被用在时间数轴 axis.type 为 'time'上,那么会被自动设置为'time'类型
'float'	如果设置成 'float',在存储时会使用 TypedArray,对性能优化有好处
'int'	如果设置成 'int',在存储时会使用 TypedArray,对性能优化有好处

【示例 3-8】 使用 dataset.dimensions 绘制折线图。

(1) 示例代码如下所示:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>the first ECharts Example</title>
        <script src="echarts.min.js"></script><!-- 引入 echarts.js -->
    </head>
    <body>
    <div id="main" style="width: 600px;height:400px;"></div>
        <script type="text/javascript">
            var myChart = echarts.init(document.getElementById('main'));
            var option = {          //指定图表的配置项和数据
                legend: {}, tooltip: {},
                dataset: {          //指定维度名顺序,利用默认的维度到坐标轴的映射
                    dimensions: [
                    {name:'product',type:'ordinal'},{name:'2023',type: 'float'},
                    {name:'2024',type: 'float'},{name:'2025',type: 'float'}],
                    source: [
                    {product:'Matcha Latte','2023':43.3,'2024':85.8,'2025':93.7},
```

```
                    {product:'Milk Tea','2023':83.1,'2024':73.4,'2025':55.1},
                    {product:'Cheese Cocoa','2023':86.4,'2024':65.2,'2025':82.5},
                    {product:'Walnut Brownie','2023':72.4,'2024':53.9,'2025':39.1} ] },
                xAxis: {type:'category'}, yAxis: {},
                series: [{type: 'line'},{type: 'line'},{type: 'line'}]
            };
            myChart.setOption(option);
        </script>
    </body>
</html>
```

(2) 运行结果如图 3-29 所示。

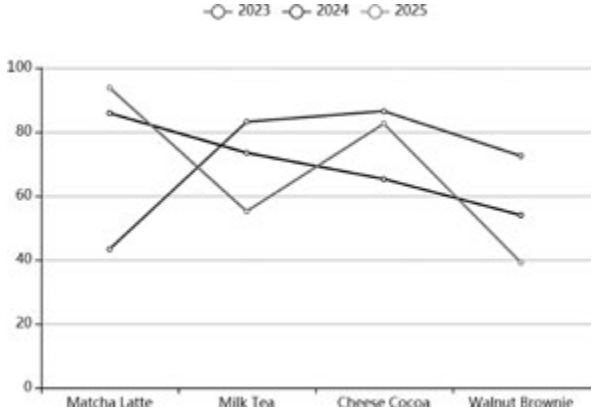


图 3-29 示例 3-8 运行结果

3.5.3 数据到图形的映射

1. series.encode 基本结构

(1) 冒号左边是坐标轴、组件等特定名称,如 x、y、tooltip 等。

(2) 冒号右边是维度名称或序号(从 0 开始计数)。

(3) 可以用数组指定多个维度。

2. series.encode 的默认值

当 series.encode 并没有指定时,Echarts 针对最常见直角坐标系中的图表(折线图、柱形图、散点图、K 线图等)、饼图、漏斗图,会采用一些默认的映射规则。规则如下。

(1) 在坐标系中(如直角坐标系、极坐标系等):

- 如果有类目轴,则将第一列(行)映射到这个轴上,后续每一列(行)对应一个系列;
- 如果没有类目轴,假如坐标系有两个轴(例如直角坐标系的 X、Y 轴),则每两列对应一个系列,这两列分别映射到这两个轴上。

(2) 如果没有坐标系(如饼图),取第一列(行)为名字,第二列(行)为数值(如果只有一列,则取第一列为数值)。

【示例 3-9】 使用 series.encode 绘制带 legend 和 tooltip 的折线图。

(1) 示例代码如下所示:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset = "utf-8">
```

```
    <title>数据到图形的映射(series.encode)</title>
    <script src = "echarts.min.js"></script>
</head>
<body>
<div id = "dom1" style = "width: 990px;height: 400px;"></div>
    <script type = "text/javascript">
        var myChart = echarts.init(document.getElementById('dom1'));
        var option = {
            dataset: {
                dimensions: ['score', 'amount', 'product'],
                source: [
                    [89.3, 58212, 'Matcha Latte'], [57.1, 78254, 'Milk Tea'],
                    [74.4, 41032, 'Cheese Cocoa'], [50.1, 12755, 'Cheese Brownie'],
                    [89.7, 20145, 'Matcha Cocoa'], [68.1, 79146, 'Tea'],
                    [19.6, 91852, 'Orange Juice'], [10.6, 101852, 'Lemon Juice'],
                    [32.7, 20112, 'Walnut Brownie'] ] },
            legend: {}, tooltip: {},
            xAxis: {type: 'value'}, yAxis: {type: 'category'},
            series: [{
                type: 'bar',
                encode: {
                    x: 0, y: 'product',  //把维度 score 和 product 映射到 x 和 y 轴
                    tooltip: [1,0, 2],  //把维度 1、维度 0 和维度 2 映射到 tooltip
        //把维度 product 和维度 1、0 连起来作为系列名,并生成相应的图例
                    seriesName: ['product', 1, 0]
                }
            }]
        };
        myChart.setOption(option);
    </script>
</body>
</html>
```

(2) 运行结果如图 3-30 所示。

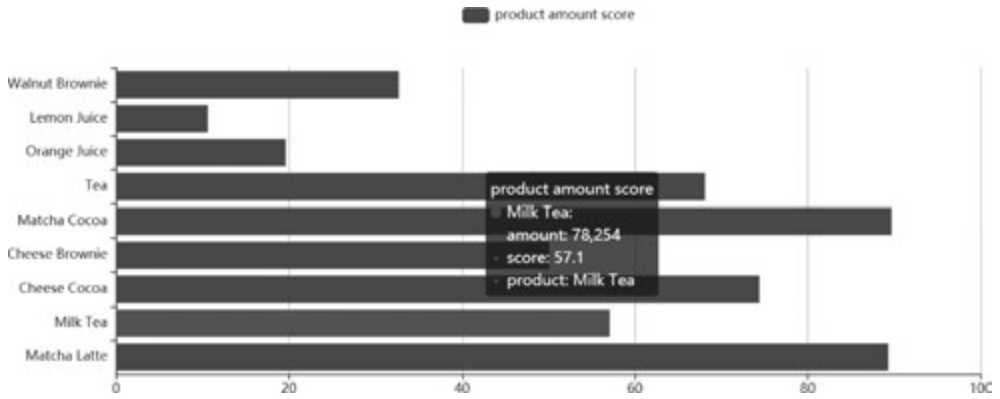


图 3-30　示例 3-9 运行结果

3.5.4　视觉通道的映射

可以使用 visualMap 组件进行视觉通道的映射。视觉元素如表 3-4 所示。

表 3-4　视觉元素

视 觉 元 素	描　　述
symbol	图元的图形类别
symbolSize	图元的大小

续表

视 觉 元 素	描　　述
color	图元的颜色
colorAlpha	图元的颜色的透明度
opacity	图元以及其附属物(如文字标签)的透明度
colorLightness	颜色的明暗度
colorSaturation	颜色的饱和度
colorHue	颜色的色调

visualMap 组件可以定义多个,从而可以同时对数据中的多个维度进行视觉映射,具体案例如下。

【示例 3-10】 通过 visualMap 实现数据通道映射。

(1) 示例代码如下所示:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>ECharts visualMap</title>
    <script src="echarts.min.js"></script><!-- 引入 echarts.js -->
    </head>
    <body>
    <div id="main" style="width: 600px;height:400px;"></div>
        <script type="text/javascript">
            var myChart = echarts.init(document.getElementById('main'));
            var option = {
                dataset: {
                    dimension: ['score', 'amount', 'product'],
                    source: [
                        [89.3, 58212, 'Matcha Latte'], [57.1, 78254, 'Milk Tea'],
                        [74.4, 41032, 'Cheese Cocoa'], [50.1, 12755, 'Cheese Brownie'],
                        [89.7, 20145, 'Matcha Cocoa'], [68.1, 79146, 'Tea'],
                        [19.6, 91852, 'Orange Juice'], [10.6, 101852, 'Lemon Juice'],
                        [32.7, 20112, 'Walnut Brownie'] ]
                },
                grid: {containLabel: true},
                xAxis: {name: 'score'}, yAxis: {type: 'category'},
                visualMap: {//Map the score column to color
                    orient: 'horizontal',
                    left: 'center',
                    min: 10,max: 100,
                    text: ['High Score', 'Low Score'],
                    dimension: 0,
                    inRange: {color: ['#65B581', '#FFCE34', '#FD665F']}
                },
                series: [{
                    type: 'bar',
                    //把维度 score 和 produc 分别映射到 x 轴和 y 轴
                    encode: { x: 'score', y: 2} }]
            };
            myChart.setOption(option);
        </script>
    </body>
</html>
```

(2) 运行结果如图 3-31 所示。

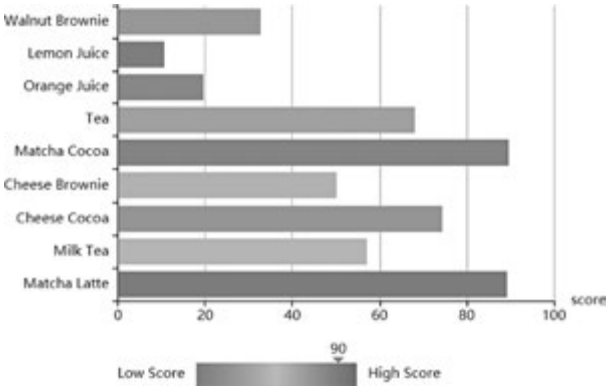


图 3-31　示例 3-10 运行结果

3.5.5　交互联动

目前并非所有图表都支持 dataset，支持 dataset 的图表有折线图、柱形图、饼图、散点图、涟漪特效动画的散点图、K 线图、地图、漏斗图等，后续会有更多的图表支持。最后，给出一个示例，多个图表共享一个 dataset，并带有联动交互。

【示例 3-11】 使用 dataset 的二维数组图表绘制。

(1) 示例代码如下：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>ECharts 实例 - 二维数组图表绘制</title>
        <script src="echarts.min.js"></script><!-- 引入 echarts.js -->
    </head>
    <body>
    <div id="main" style="width: 600px;height:400px;"></div>
        <script type="text/javascript">
            var myChart = echarts.init(document.getElementById('main'));
            setTimeout(function() {      //在指定的毫秒数后调用函数或计算表达式
                var option = {
                    legend: {},
                    tooltip: {          //x 轴触发,不显示提示框
                        trigger: 'axis', showContent: false },
                    dataset: {
                        source:
                        [['product', '2020', '2021', '2022', '2023', '2024', '2025'],
                        ['Matcha Latte', 41.1, 30.4, 65.1, 53.3, 83.8, 98.7],
                        ['Milk Tea', 86.5, 92.1, 85.7, 83.1, 73.4, 55.1],
                        ['Cheese Cocoa', 24.1, 67.2, 79.5, 86.4, 65.2, 82.5],
                        ['Walnut Brownie', 55.2, 67.1, 69.2, 72.4, 53.9, 39.1]]},
                    xAxis: {type: 'category'}, yAxis: {gridIndex: 0},
                    grid: {top: '55%',containLabel: true},
                    series:[{type: 'line',smooth: true,seriesLayoutBy:'row'},
                            {type: 'line',smooth: true,seriesLayoutBy: 'row'},
                            {type: 'line',smooth: true,seriesLayoutBy: 'row'},
                            {type: 'line',smooth: true,seriesLayoutBy: 'row'},
                            {type: 'pie',id: 'pie',radius: '30%',center: ['50%', '25%'],
                    //{a}(系列名称),{b}(数据项名称),{c}(数值), {d}(百分比)
                    label: {formatter: '{b}: {@2012} ({d}%)'},
                    encode: {itemName: 'product',value: '2012',tooltip: '2012'}}
```

```
                    ], };
                //通过鼠标事件,不断获取 xAxisInfo
                //然后根据获取到的 xAxisInfo.value(dimension)重新绘制饼图
                myChart.on('updateAxisPointer', function(event) {
                    var xAxisInfo = event.axesInfo[0];
                    if (xAxisInfo) {
                        var dimension = xAxisInfo.value + 1;
                        myChart.setOption({
                            series: {
                            id: 'pie',
                            label: {formatter: '{b}: {@[' + dimension + ']} ({d}%)'},
                            encode: {value: dimension,tooltip: dimension}}
                        });
                    }
                });
                myChart.setOption(option);
            });
        </script>
    </body>
</html>
```

(2) 运行结果如图 3-32 所示,饼图数据和区域会随着鼠标在折线图上位置的变化而变化。

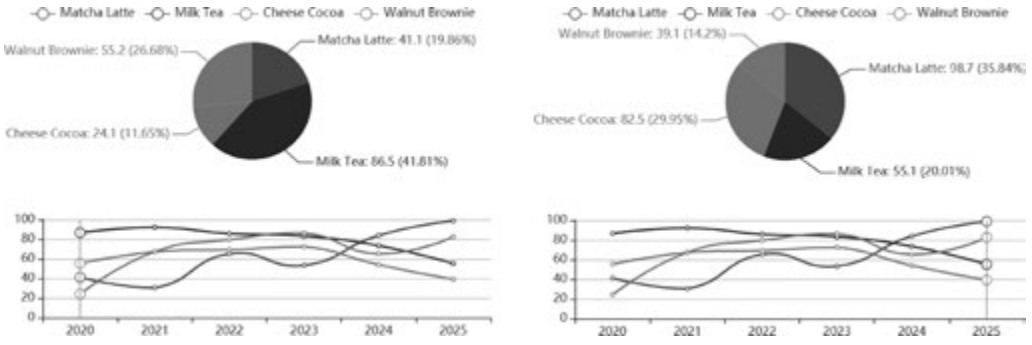


图 3-32 示例 3-11 运行结果

3.6 Echarts 交互组件


视频讲解

Echarts 提供了很多交互组件,如图例组件 legend、标题组件 title、视觉映射组件 visualMap、数据区域缩放组件 dataZoom、时间线组件 timeline。dataZoom 组件可以实现通过鼠标滚轮滚动放大缩小图表的功能。

1. 拖动 dataZoom 数据区组件

dataZoom 组件通过拖动 dataZoom 组件可实现缩放和平移图表的数据窗口功能。默认下 dataZoom 控制 x 轴,即对 x 轴进行数据窗口平移操作。dataZoom 组件有 inslide 和 slide 两种类型,分别表示有滑动块的和内置的。在如下的示例中通过拖动 dataZoom 组件控制 x 轴,即对 x 轴进行数据窗口缩放和数据窗口平移操作。

【示例 3-12】 拖动 dataZoom 组件控制 x 轴,实现数据区域的平移和缩放。

(1) 示例代码如下:

```
<!DOCTYPE html>
<html>
```

```
    <head>
        <meta charset="utf-8">
        <title>ECharts 实例 - 拖动 dataZoom 组件</title>
        <script src="echarts.min.js"></script> <!-- 引入 echarts.js -->
    </head>
    <body>
    <div id="dom" style="width: 600px;height:400px;"></div>
          <script type="text/javascript">
            var myChart = echarts.init(document.getElementById('dom'));
            var option = {                    //指定图表的配置项和数据
                title: {        //默认为'left',可选'center','left','right'或者 x 坐标,单位 px
                    text: 'scatter',x: 'center',y: 'top'},
                     xAxis: {type: 'value'},yAxis: {type: 'value'},
                     tooltip: {},
                dataZoom: [{                  //这个 dataZoom 组件,默认控制 x 轴
                    type: 'slider',           //dataZoom 组件是 slider 型
                    start: 10,                //左边在 10 % 的位置
                    end: 60                   //右边在 60 % 的位置
                },                            //这个 dataZoom 组件,也控制 x 轴
                {
                type: 'inside',     //这个 dataZoom 组件是 inside 型 dataZoom 组件
                start: 10,end: 60             //左边在 10 %,右边在 60 % 的位置
                }, ],
                series: [                     //类别为散点图,透明度为 0.8
                {type: 'scatter', itemStyle: {opacity: 0.8},
                symbolSize: function(val) {   //使用回调函数设置散点的大小
                    return val[2] * 40;
                },
                data: [
                ["14.616", "7.241", "0.896"], ["3.958", "5.701", "0.955"],
                ["2.768", "8.971", "0.669"], ["9.051", "9.710", "0.171"],
                ["14.046", "4.182", "0.536"], ["12.295", "1.429", "0.962"],
                ["4.417", "8.167", "0.113"], ["0.492", "4.771", "0.785"],
                ["7.632", "2.605", "0.645"], ["14.242", "5.042", "0.368"]],
                tooltip: {                    //提示信息为鼠标悬停位置的横纵坐标
                    formatter: function(params) {
                        return params.data[0] + "," + params.data[1];}
                    }
                }, ]
            }
            myChart.setOption(option);
        </script>
    </body>
</html>
```

(2) 运行结果如图 3-33 所示。

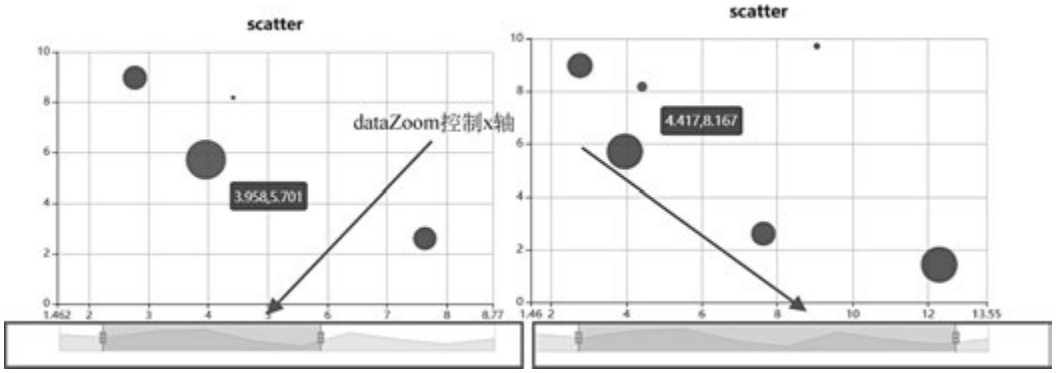


图 3-33 示例 3-12 运行结果

2. 在数据区滚动鼠标

上面的实例只能拖动 dataZoom 组件来缩小或放大图表。如果想在坐标系内进行拖动，以及用鼠标滚轮（或移动触屏上的两指滑动）进行缩放，那么需要再加上一个 inside 型的 dataZoom 组件。在以上实例基础上我们再增加 type:'inside'的配置信息，代码如图 3-34 所示，用户不仅可以拖动 dataZoom 的滑条，也可以在图表区域内滚动鼠标滚轮来实现同样的功能。

```
var option = {
        xAxis: {type: 'value'},
        yAxis: {type: 'value'},
        tooltip:{},
        dataZoom: [{// 这个dataZoom组件，默认控制x轴
            type: 'slider', //dataZoom组件是slider型
            start: 10,      // 左边在 10% 的位置
            end: 60         // 右边在 60% 的位置
        },
        {   // 这个dataZoom组件，也控制x轴
            type: 'inside', // 这个 dataZoom 组件是 inside 型 dataZoom 组件
            start: 10,      // 左边在 10% 的位置
            end: 60         // 右边在 60% 的位置
        }],
```

图 3-34 使用 dataZoom 和鼠标滚轮控制 x 轴

3.7 Echarts 响应式

Echarts 图表显示在用户指定宽高的 DOM 节点（容器）中。如果想在计算机和移动设备上很好地展示图表的内容，实现响应式的设计，就需要 Echarts 完善组件的定位设置，并且实现类似 CSS Media Query 的自适应能力。

3.7.1 Echarts 组件的定位和布局

大部分组件和系列会遵循以下两种定位方式。

1. left/right/top/bottom/width/height 定位方式

这 6 个量中，每个量都可以是绝对值，或者百分比，或者位置描述。

（1）绝对值：单位是浏览器像素（px），用 number 形式书写（不写单位）。例如{left:23，height:400}。

（2）百分比：表示占 DOM 容器宽高的比例，用 string 形式书写。例如{right:'30%'，bottom:'40%'}。

（3）位置描述：可以设置 left:'center'，表示水平居中；可以设置 top:'middle'，表示垂直居中。

2. center / radius 定位方式

在自适应容器大小时，百分比设置是很有用的。

（1）center：是一个数组，表示[x，y]，其中，x、y 可以是绝对值或者百分比，含义和前述相同。

（2）radius：是一个数组，表示[内半径，外半径]，其中，内外半径可以是绝对值或者百分比，含义和前述相同。

在自适应容器大小时，百分比设置是很有用的。

3. 横向（horizontal）和纵向（vertical）

Echarts 外观狭长形的组件（如 legend、visualMap、dataZoom、timeline 等），大多提供了横向布局或纵向布局的选择。例如，在细长的移动端屏幕上，适合使用纵向布局；在 PC 宽屏上，适合使用横向布局。横纵向布局的设置，一般在组件或者系列的 orient 或者 layout 配置项

上,设置为 'horizontal' 或者 'vertical'。

3.7.2 Media Query 自适应

Media Query 提供了组件随着容器尺寸改变而改变的能力。以下这个例子,可尝试拖动右下角的圆点,随着尺寸变化,legend 和系列会自动改变布局位置和方式,如图 3-35 所示。

【示例 3-13】 组件会随着屏幕尺寸变化,legend 和系列会自动改变布局位置和方式。

(1) 示例代码如下所示:

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>ECharts 实例</title>
    <script src="jquery.min.js"></script>
  <script src="echarts.min.js"></script> <!-- 引入 echarts.js -->
</head>
<body>
  <!-- 为 ECharts 准备一个具备大小(宽高)的 DOM -->
  <div id="main" style="width: 600px;height:400px;"></div>
  <script type="text/javascript">
    //基于准备好的 DOM,初始化 Echarts 实例
    var myChart = echarts.init(document.getElementById('main'));
     $.when(
       $.getScript('https://www.runoob.com/static/js/timelineGDP.js'),
       $.getScript('https://www.runoob.com/static/js/draggable.js')
    ).done(function () {
        draggable.init(
           $('div[_echarts_instance_]')[0],
          myChart,{width: 700, height: 400,throttle: 70});
        myChart.hideLoading();
        option = {
          baseOption:{
            title: {text: '南丁格尔玫瑰图',subtext: '纯属虚构',x:'center'},
            tooltip:{trigger: 'item',formatter: "{a} <br/>{b} : {c} ({d}%)"},
            legend:
            {data:['rose1','rose2','rose3','rose4','rose5','rose6','rose7','rose8']},
            toolbox: {
              show : true,
              feature : {
              mark : {show: true},
              dataView : {show: true, readOnly: false},
              magicType : {how: true,type: ['pie', 'funnel']},
              restore : {show: true},
              saveAsImage : {show: true}}
            },
            calculable : true,          //启用拖曳重计算特性
            series : [
            {name:'半径模式', type:'pie',roseType : 'radius',
            label: {normal: {show: false},emphasis: {show: true}},
            lableLine: {normal: {show: false},
            emphasis: {show: true}},
            data:[
            {value:10, name:'rose1'},{value:5, name:'rose2'},
            {value:15, name:'rose3'},{value:25, name:'rose4'},
            {value:20, name:'rose5'},{value:35, name:'rose6'},
            {value:30, name:'rose7'}, {value:40, name:'rose8'}]
          },
          {name:'面积模式',type:'pie',roseType : 'area',
```

```
            data:[
            {value:10, name:'rose1'},{value:5, name:'rose2'},
            {value:15, name:'rose3'}, {value:25, name:'rose4'},
            {value:20, name:'rose5'}, {value:35, name:'rose6'},
            {value:30, name:'rose7'}, {value:40, name:'rose8'}]}
        ]},
        media: [{
          option: {
            legend: {right: 'center',bottom: 0,orient: 'horizontal'},
            series: [                              //系列的位置
            {radius: [20, '50%'],center: ['25%', '50%']},
            {radius: [30, '50%'],center: ['75%', '50%']}]}},
            {query: {minAspectRatio: 1},           //当长宽比大于或等于 1
            option: {                              //此规则满足下的 option
              legend:{right: 'center', bottom: 0,orient: 'horizontal'},
              series: [
              {radius: [20, '50%'],center: ['25%', '50%']},
              {radius: [30, '50%'],center: ['75%', '50%']}]}},
              {query: {maxAspectRatio: 1},         //当长宽比小于或等于 1
              option: {
                legend:
                {right: 'center',bottom: 0,orient: 'horizontal'},
                series: [
                {radius: [20, '50%'],center: ['50%', '30%']},
                {radius: [30, '50%'],center: ['50%', '70%']}]}
              },
              {query: { maxWidth: 500},
              option: {
                legend:
                {right: 10,top: '15%',orient: 'vertical'},
                series: [
                {radius: [20, '50%'],center: ['50%', '30%']},
                {radius: [30, '50%'], center: ['50%', '75%']}]
                }
              }
            ]
          };
          myChart.setOption(option);
        });
    </script>
  </body>
  </html>
```

(2) 运行结果如图 3-35 所示,其中图 3-35(a)为默认规则下图表和图例布局；图 3-35(b)为长宽比大于或等于 1 时图表和图例布局；图 3-35(c)为长宽比小于或等于 1 时图表和图例布局；图 3-35(d)为窗口最大宽度小于 500 时图表和图例布局。

对于图表、legend 等元素的自适应实现,必须在 option 中设置 Media Query 遵循如下格式：

```
option = {                      //这里是基本的“原子 option”
  title: {...},
  legend: {...},
  series: [{...}, {...}, ...], ...,
  media: [                      //这里定义了 media query 的逐条规则
    {
      query: {...},             //这里写规则
      option: {                 //这里写此规则满足下的 option
        legend: {...}, ... } },
    {
      query: {...},             //第二个规则
```

```
        option: {                    //第二个规则对应的 option
          legend: {...}, ... } },
      {                              //这条里没有写规则,表示"默认"
        option: {                    //即所有规则都不满足时,采纳这个 option
          legend: {...},  ... }
      }
    ]
};
```

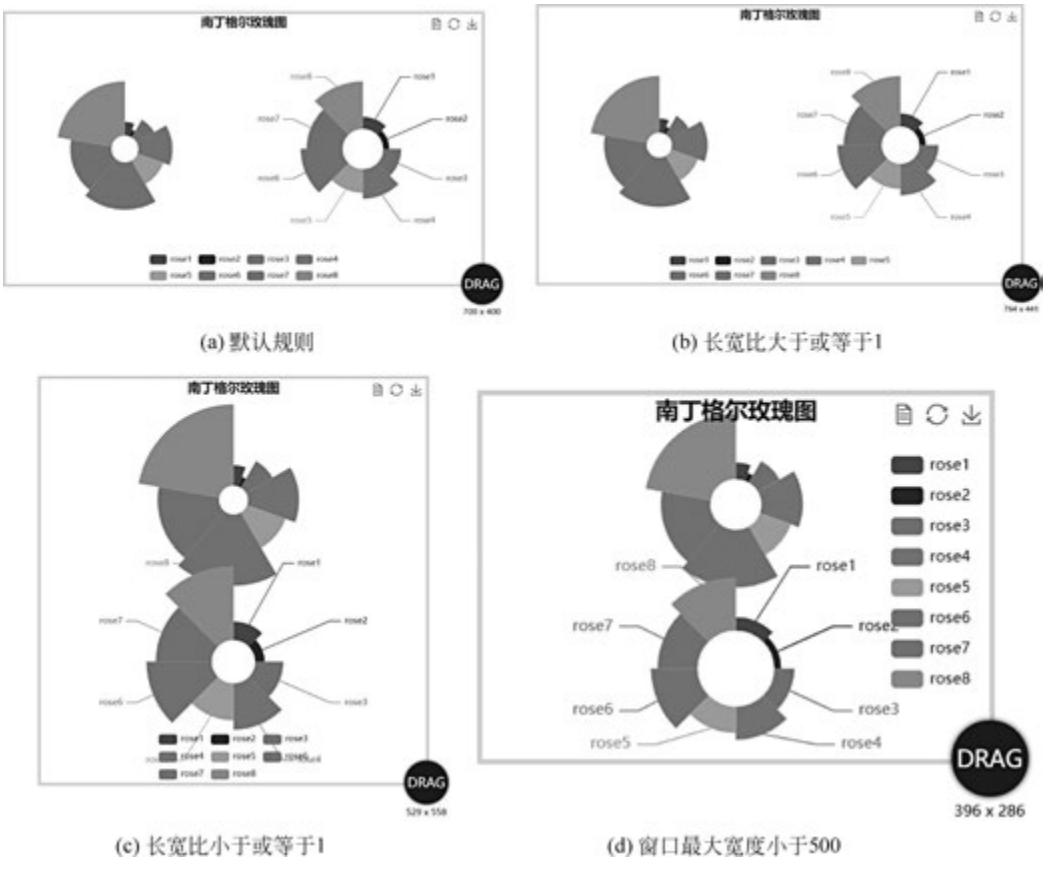


图 3-35　示例 3-13 运行结果

示例 3-13 中,baseOption 以及 media 的每个 option 都是原子 option,即包含各组件、系列定义的 option。而由原子 option 组合成的整个 option,我们称为复合 option。我们通常使用 baseOption 来设置原子 option;此外,满足了某个 query 条件时,对应的 option 会被使用 chart.mergeOption()来 merge(合并)进去。

下面介绍 Media Query 自适应的各个参数的设置和注意事项。

1. query

query 现在支持三个属性:width、height、aspectRatio(长宽比)。每个属性都可以加上 min 或 max 前缀。比如,minWidth:200 表示大于或等于 200px 宽度。两个属性一起写表示"并且",比如:{minWidth:200, maxHeight:300} 表示大于或等于 200px 宽度,并且小于或等于 300px 高度。基本的 query 设置如下:

```
{
  minWidth:200, maxHeight:300,
```

```
    minAspectRatio:1.3
}
```

2. option

media 中的 option 是原子 option，理论上可以写任何 option 的配置项。但是一般我们只写和布局定位相关的部分，例如截取示例 3-13 中的一部分 query option：

```
media: [
  ...,
  {
    query: {
      maxAspectRatio: 1                    //当长宽比小于 1 时
    },
    option: {
      legend: {                            //legend 放在底部中间
        right: 'center',
        bottom: 0,
        orient: 'horizontal'               //legend 横向布局
      },
      series: [                            //两个饼图左右布局
        {radius: [20, '50%'],center: ['50%', '30%']},
        {radius: [30, '50%'],center: ['50%', '70%']}]
    }
  },{
    query: {
      maxWidth: 500                        //当容器宽度小于 500 时
    },
    option: {
      legend: {
        right: 10,                         //legend 放置在右侧中间
        top: '15%',
        orient: 'vertical'                 //纵向布局
      },
      series: [                            //两个饼图上下布局
        {radius: [20, '50%'],center: ['50%', '30%']},
        {radius: [30, '50%'],center: ['50%', '75%']
        }
      ]
    }
  }, ...
]
```

3. 多个 query 被满足时的优先级

可以有多个 query 同时被满足，都会被 mergeOption，定义在后面的 query 的优先级更高，即定义在后面的会先被 merge。

4. 默认 query

如果 media 中有某项不写 query，则表示默认值，即所有规则都不满足时，采纳这个 option。

5. 容器大小实时变化时的注意事项

在多数情况下，并不需要容器 DOM 节点任意随着拖曳变化大小，而只是根据不同终端设置几个典型尺寸。但是如果容器 DOM 节点需要能任意随着拖曳变化大小，那么目前使用时需要注意：某个配置项，如果在某一个 query option 中出现，那么在其他 query option 中也必须出现，否则不能回归到原来的状态；但是 left、right、top、bottom、width 和 height 不受这个限制。

6. 复合 option 中的 media 不支持 merge

也就是说，当第二（或三、四、五、……）次 chart.setOption(rawOption)时，如果 rawOption 是

复合 option(即包含 media 列表),那么新的 rawOption. media 列表不会和旧的 media 列表进行 merge,而是简单替代。当然,baseOption 仍然会正常和旧的 option 进行 merge。

其实,很少有场景需要使用复合 option 来多次 setOption,推荐的方法是,使用 mediaQuery 时,第一次 setOption 使用复合 option,后面 setOption 时仅使用原子 option,也就是仅用 setOption 来改变 baseOption。

3.8 Echarts 数据视觉映射

数据可视化是数据到视觉元素的映射过程(这个过程也可称为视觉编码,视觉元素也可称为视觉通道)。

Apache Echarts™ 的每种图表都内置了这种映射过程,比如折线图把数据映射到线,柱形图把数据映射到长度。一些更复杂的图表,如事件河流图、树图也都会做出它们内置的映射。

此外,Echarts 还提供了 visualMap 组件来提供通用的视觉映射。visualMap 组件中可以使用的视觉元素有:图形类别(symbol)、图形大小(symbolSize)、颜色(color)、透明度(opacity)、颜色透明度(colorAlpha)、颜色明暗度(colorLightness)、颜色饱和度(colorSaturation)、色调(colorHue)。

下面对 visualMap 组件的使用方式进行简要的介绍。

3.8.1 数据和维度

Echarts 中的数据一般存放于 series. data 中。根据图表类型不同,数据的具体形式也可能有些许差异。比如可能是线性表、树、图等。但它们都有个共性:都是数据项(dataItem)的集合。每个数据项含有数据值(value)和其他信息(如果需要的话)。每个数据值,可以是单一的数值(一维)或者一个数组(多维)。

例如,series. data 最常见的形式是线性表,即一个普通数组,代码如下所示:

```
series: {
  data: [                                  //这里每一项都是数据项(dataItem)
    {
      value: 2323,                         //这是数据项的数据值(value)
      itemStyle: {...}
    },
    //也可以直接是 dataItem 的 value,这更常见,每个 value 都是一维
    1212, 2323, 4343, 3434]
}
series: {
  data: [                                  //这里每一项都是数据项(dataItem)
    {
      value: [3434, 129, '圣马力诺'],      //这是数据项的数据值(value)
      itemStyle: {...}
    },
    [1212, 5454, '梵蒂冈'],                //也可以直接是 dataItem 的 value,这更常见
    [2323, 3223, '瑙鲁'],                  //每个 value 都是三维的,每列是一个维度
    [4343, 23, '图瓦卢']                   //假如是气泡图,常见第一维度映射到 x 轴,
                                           //第二维度映射到 y 轴,
                                           //第三维度映射到气泡半径(symbolSize)
  ]
}
```

在图表中，往往默认把 value 的前一两个维度进行映射，比如取第一个维度映射到 x 轴，取第二个维度映射到 y 轴。如果想要把更多的维度展现出来，可以借助 visualMap。

3.8.2 visualMap 组件

visualMap 组件可以把数据的指定维度映射到对应的视觉元素上。visualMap 组件可以定义多个，从而可以同时对数据中的多个维度进行视觉映射。visualMap 组件可以定义为分段型(visualMapPiecewise)或连续型(visualMapContinuous)，通过 type 来区分。

```
option = {
visualMap: [
  {                        //第一个 visualMap 组件
    type: 'continuous',    //定义为连续型 visualMap
     ... },
  {                        //第二个 visualMap 组件
     type: 'piecewise',    //定义为分段型 visualMap
    ... }
  ]
  ...
};
```

连续型是指进行视觉映射的数据维度是连续的数值；而分段型则是数据被分成了多段或者是离散型的数据。

1. 连续型视觉映射

连续型视觉映射通过指定最大值、最小值来确定视觉映射的范围。

```
option = {
      visualMap: [
        {
         type: 'continuous',
         min: 0,
         max: 5000,
         dimension: 3,                    //series.data 的第四个维度(即 value[3])被映射
         seriesIndex: 4,                  //对第四个系列进行映射
         inRange: {                       //选中范围中的视觉配置
          //数据最小值映射到'blue'上,最大值映射到'red'上,其余自动线性计算
          color: ['blue', '#121122', 'red'],       //定义了图形颜色映射的颜色列表
          //数据最小值映射到 30 上,最大值映射到 100 上,其余自动线性计算
          symbolSize: [30, 100]                    //定义了图形尺寸的映射范围
         },
         outOfRange: {
          //选中范围外的视觉配置
          symbolSize: [30, 100]
         }
        }
        //  ...
      ]
};...
```

其中，visualMap. inRange 表示在数据映射范围内的数据采用的样式；visualMap. outOfRange 指定超出映射范围外的数据的样式；visualMap. dimension 指定将数据的哪个维度做视觉映射。

2. 分段型视觉映射

分段型视觉映射组件有以下 3 种模式。

(1) 连续型数据平均分段：依据 visualMap-piecewise. splitNumber 来自动平均分割成若干块。

(2) 连续型数据自定义分段：依据 visualMap-piecewise. pieces 来定义每块范围。

(3) 离散数据(类别性数据)：类别定义在 visualMap-piecewise. categories 中。

使用分段型视觉映射时，需要将 type 设为 'piecewise'，并且选择上面 3 个配置项中的一个，其他配置项类似连续型视觉映射。