1.数据可视化前言

1.1.什么是数据可视化

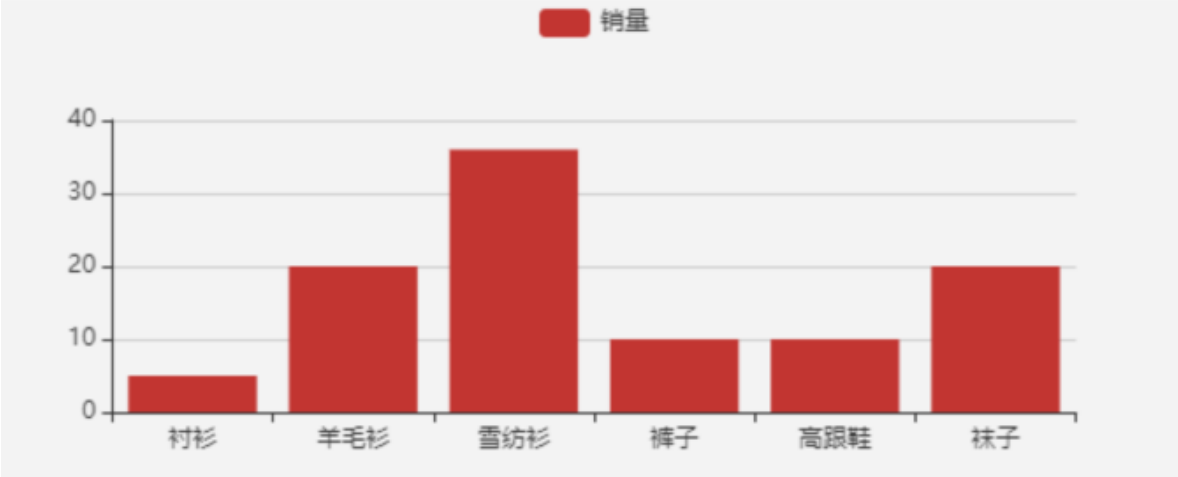
数据可视化, 说白了, 就是把数据以更加直观的方式进行呈现. 那什么方式是更加直观的方式呢? 就是图表.

常言道, 文不如表, 表不如图, 人们大脑对图的敏感程度要比苍白无力的文字好很多.

我们来看一组数据.

```
衬衫:5
羊毛衫:20
雪纺衫:36
裤子:10
高跟鞋:10
袜子:20
```

这个数据就是某些产品的销量. 单纯从这些文字上来看, 很难看出数据之间对比的关系. 如果把这些数据以图表的方式呈现出来呢？

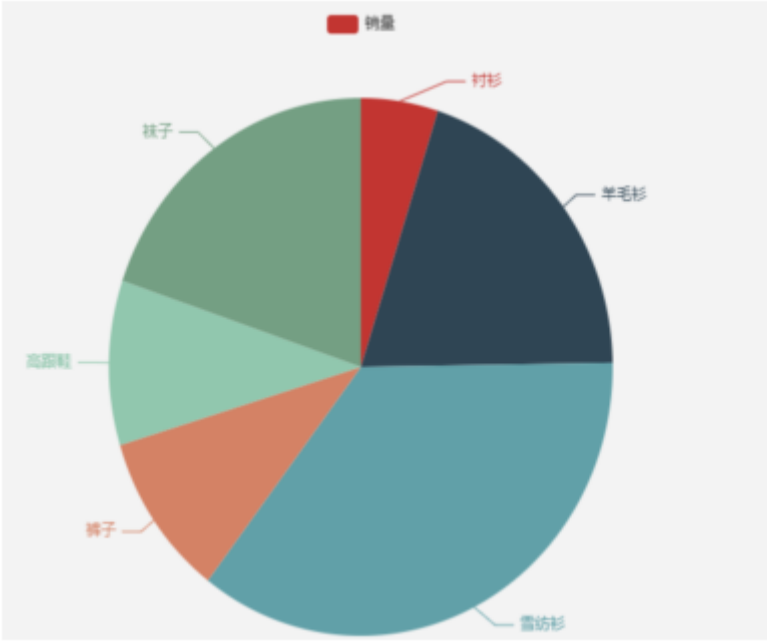


上面这幅图就是这组数据的图表展示. 通过这幅图一眼就能看出哪些产品销量高, 哪些产品销量低. 数据与数据之间的关系一目了然.

1.2.数据可视化的好处

- 清晰有效地传达与沟通信息

  数据可视化的好处之一就是能够清晰有效的传达信息和沟通信息. 继续看刚才的那个例子, 如果使用同样的数据, 换成另外一种展现形式, 比如下边的这幅饼图. 我们可以很容易的就看出每个产品的销量占比.不需要太多的脑力计算和思维转换.



- 更容易洞察隐藏在数据中的信息

  将数据以图表的方式呈现出来还可以帮助我们感受到那些隐藏在数据之间的信息.比如下面的这幅上证指数的k线图

  这幅图中可以看出指数的上升趋势或者下降趋势. 而上升趋势或者下降趋势这种信息是很难从文字中察觉到.

1.3.数据可视化的实现方式

- 报表类
  - `Excel`
  - 水晶报表

> 报表类的主要实现方式就大家熟悉的Excel或者水晶报表， 这种方式主要面向的是非技术人员， 在特定的软件中点击几个按钮,添加一些数据就可以生成图标了.这种方式的优点是简单， 谁都会用. 缺点也显而易见， 就是不灵活， 图表一旦生成之后就固定不变了， 如果数据发生变化了， 图表需要重新生成

- 商业智能 BI
  - Microsoft BI
  - Power-BI

> 商业智能BI的实现方式主要有微软的BI和Power-BI，它比报表类更加高端，他除了可以对数据生成报表之外，还可以提出决策依据，帮助企业做出明智的业务经营决策

- 编码类
  - ECharts.js
  - D3.js

> 编码类，这种是需要程序员参与，程序员可以对接到公司现有的系统架构中进行编码，实时生成动态的图表.常见的使用库有ECharts.js和D3.js，我们项目中使用的是ECharts.js ，他是百度公司开发的一套开源可视化库，D3.js是国外的一个可视化库，在封装性\易用性\效果上，ECharts要更优秀一些.

相对来说,这三种方式中编码类的实现方式更加灵活, 他可以融入到我们已有的项目中,和项目的贴合度是最高的, 但是他的门槛也高些, 需要有编程基础才能完成. 而我们的这么课程正是编码类可视化的实现, 并且选择的是百度开源的 ECharts.js .

2. ECharts 的基本使用

2.1. ECharts 的介绍

ECharts是百度公司开源的一个使用 JavaScript 实现的开源可视化库，兼容性强，底层依赖矢量图形库 ZRender，提供直观，交互丰富，可高度个性化定制的数据可视化图表。

- 开源免费

  它是开源免费的，也就是我们可以免费的使用 ECharts，不需要缴纳任何的费用

- 功能丰富

  它的功能非常的丰富，提供了各种各样的图表，支持各种各样的定制, 满足各种需求，比如折线图、柱状图、饼图、K线图等. 在他的官方示例中, 提供了上百种图表, 可以用 只有你想不到, 没有她做不到 这句话来形容

- 社区活跃

  ECharts 的社区非常活跃，意味着你可以和很多开发者讨论，遇到了 ECharts 中不会的问题，也很容易找到解决办法

- 多种数据的支持

  可视化的含义就是将数据通过更加直观的图表的方式来呈现。图表只是一种呈现方式。最核心的其实是数据。 ECharts 对数据格式的支持也是非常友好的。 ECharts 能够支持常见的 key-value 数据格式，还能支持二维表，或者 TypedArray 格式的数据

- 流数据的支持

  对于超大的数据量而言， 数据本身的体量可能就非常消耗资源, 而 ECharts 可以支持对流数据的动态渲染，加载多少数据就渲染多少数据，省去了漫长的数据加载的等待时间, 他还提供了增量渲染的技术, 只渲染变化的数据, 提高系统的资源利用.

- 移动端的优化
- 跨平台

- 酷炫的特效,
- 数据的三维可视化
- ……

> ECharts 能够做出各种各样漂亮的图表，它能满足绝大多数可视化图表的实现.它的兼容性强， 使用方便，功能强大， 是实现数据可视化的最佳选择之一， 更多特点和介绍可以查阅官网地址：
> https://echarts.apache.org/zh/index.html

2.2.ECharts的快速上手

ECharts的入门使用特别简单, 5分钟就能够上手. 他大体分为这几个步骤

- 步骤1：引入echarts.js文件

  echarts是一个js的库，当然得先引入这个库文件

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

- 步骤2：准备一个呈现图表的盒子

  这个盒子通常来说就是我们熟悉的div，这个div决定了图表显示在哪里

  ```
  <div id="main" style="width: 600px;height:400px;"></div>
  ```

- 步骤3：初始化echarts实例对象

  在这个步骤中, 需要指明图表最终显示在哪里的DOM元素

  ```
  var myChart = echarts.init(document.getElementById('main'))
  ```

- 步骤4：准备配置项

  这步很关键，我们最终的效果，到底是显示饼图还是折线图，基本上都是由配置项决定的

  ```
  var option = {
      xAxis: {
          type: 'category',
          data: ['小明', '小红', '小王']
      },
      yAxis: {
          type: 'value'
      },
      series: [
          {
              name: '语文',
              type: 'bar',
              data: [70, 92, 87],
          }
      ]
  }
  ```

- 步骤5：将配置项设置给echarts实例对象

  ```
  myChart.setOption(option)
  ```

通过简单的5个步骤, 就能够把一个简单的柱状图给显示在网页中了.这几个步骤中, 步骤4最重要,

一个图表最终呈现什么样子,完全取决于这个配置项.所以对于不同的图表, 除了配置项会发生改变之外, 其他的代码 都是固定不变的.

2.3.相关配置讲解

- `xAxis`

  直角坐标系 中的 `x` 轴, 如果 `type` 属性的值为 `category` ,那么需要配置 `data` 数据, 代表在 `x` 轴的呈现

- `yAxis`

  直角坐标系 中的 `y` 轴, 如果 `type` 属性配置为 `value` , 那么无需配置 `data` , 此时 `y` 轴会自动去 `series` 下找数据进行图表的绘制

- `series`

  系列列表。每个系列通过 `type` 决定自己的图表类型, `data` 来设置每个系列的数据

配置项都是以键值对的形式存在, 并且配置项有很多, `ECharts` 的学习大多是针对于这些配置项的, 对于配置项的学习, 大家可以不用死记硬背, 需要的时候查一查官方文档即可. 网址: `https://echarts.apache.org/zh/option.html` , 常用的配置项多用几次, 你自然而然就记下了

同学们可以查文档试一下: title中的各种配置

```
title: {
    show: true,
    text: '标题',
    link: 'http://www.baidu.con',
    textStyle: {
        color: 'red'
    }
}
```

-

3.`ECharts`常用图表

3.1.图表1 柱状图

3.1.1.柱状图的实现步骤

- 步骤1 `ECharts` 最基本的代码结构

```
<!DOCTYPE html>
<html lang="en">
<head>
    <script src="js/echarts.min.js"></script>
</head>
<body>
    <div style="width: 600px;height:400px"></div>
    <script>
        var mCharts = echarts.init(document.querySelector("div"))
        var option = {}
```

```
        mCharts.setOption(option)
    </script>
</body>
</html>
```

此时`option`是一个空空如也的对象

- 步骤2 准备x轴的数据

```
var xDataArr = ['张三', '李四', '王五', '闰土', '小明', '茅台', '二妞', '大强']
```

- 步骤3 准备`y`轴的数据

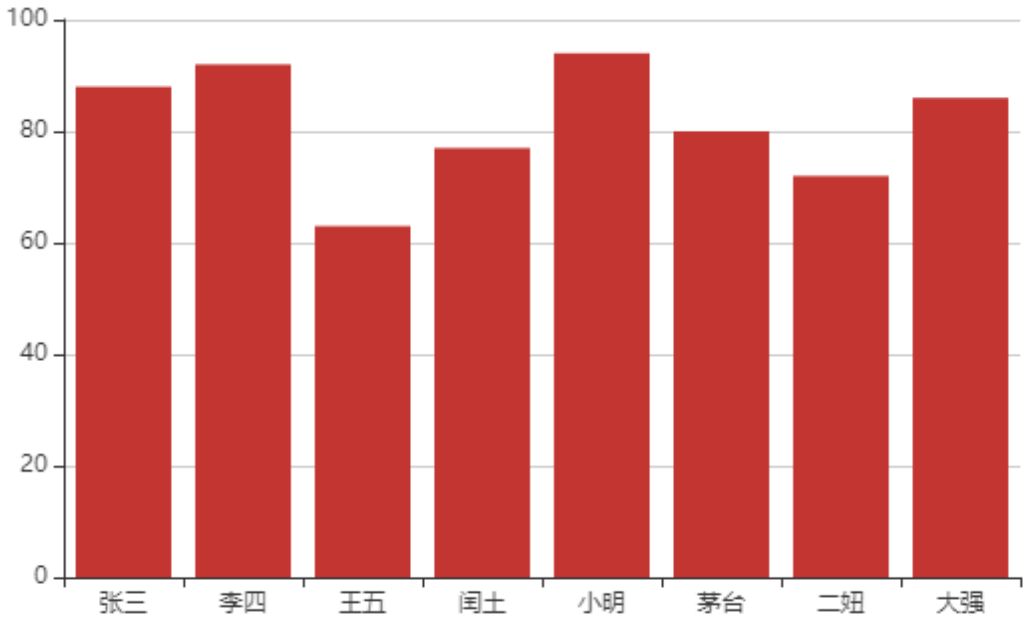
```
var yDataArr = [88, 92, 63, 77, 94, 80, 72, 86]
```

- 步骤4 准备`option`, 将`series`中的`type`的值设置为: `bar`

```
var option = {
    xAxis: {
        type: 'category',
        data: xDataArr
    },
    yAxis: {
        type: 'value'
    },
    series: [
        {
            type: 'bar',
            data: yDataArr
        }
    ]
}
```

注意: 坐标轴`xAxis`或者`yAxis`中的配置, `type`的值主要有两种: `category`和`value`, 如果`type`属性的值为`category`,那么需要配置`data`数据, 代表在`x`轴的呈现. 如果`type`属性配置为`value`, 那么无需配置`data`, 此时`y`轴会自动去`series`下找数据进行图表的绘制

最终的效果如下图:



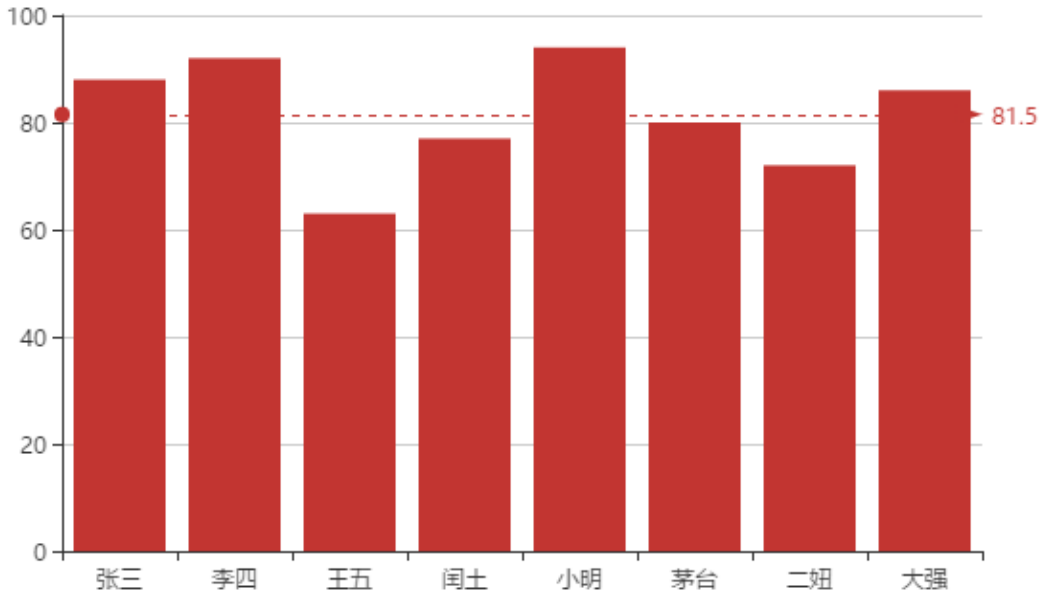
3.1.2.柱状图的常见效果

- 标记:
  - 最大值\最小值 `markPoint`

```
series: [
    {
        ......
        markPoint: {
            data: [
                {
                    type: 'max', name: '最大值'
                },
                {
                    type: 'min', name: '最小值'
                }
            ]
        }
    }
]
```

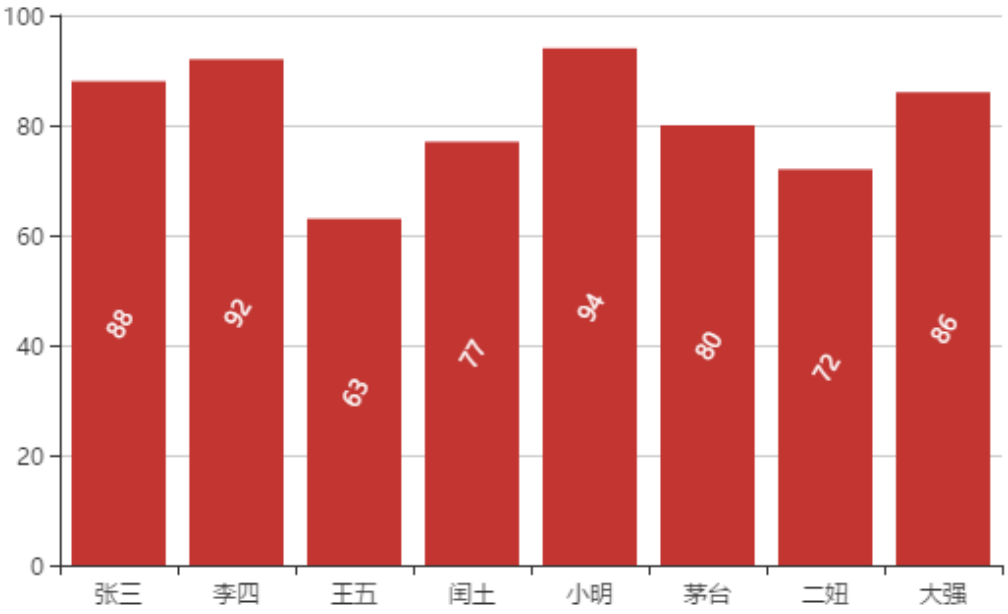
- 平均值 `markLine`

```
series: [
    {
        ......
        markLine: {
            data: [
                {
                    type: 'average', name: '平均值'
                }
            ]
        }
    }
]
```



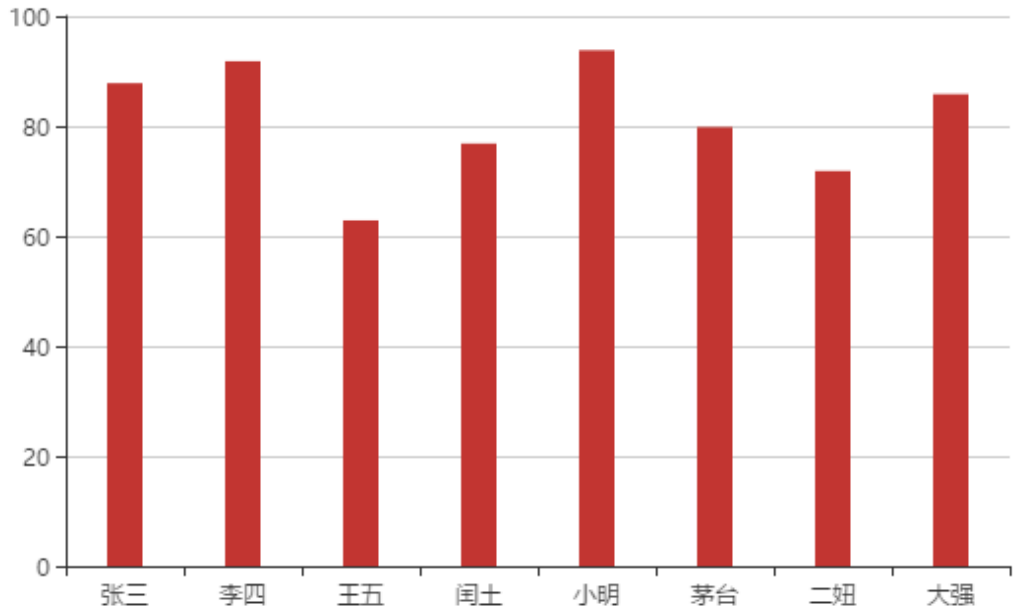
- 显示
  - 数值显示 `label`

```
series: [
    {
        ......
        label: {
            show: true, // 是否可见
            rotate: 60 // 旋转角度
        }
    }
]
```



- 柱宽度 `barWidth`

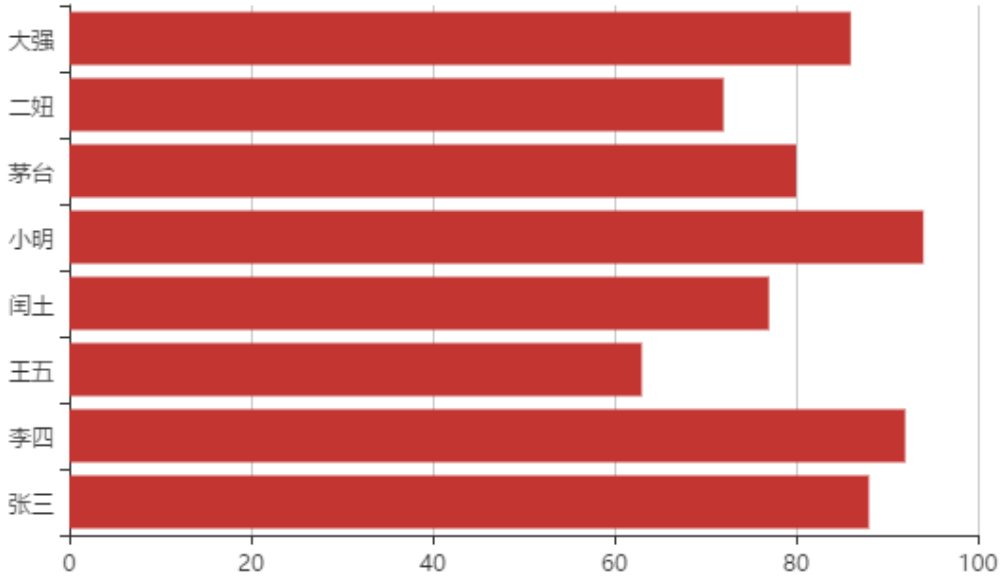
```
series: [
    {
        ......
        barWidth: '30%' // 柱的宽度
    }
]
```



- 横向柱状图

  所谓的横向柱状图, 只需要让x轴的角色和y轴的角色互换一下即可. 既 `xAxis` 的 `type` 设置为 `value`, `yAxis` 的 `type` 设置为 `category`, 并且设置 `data` 即可

```
var option = {
    xAxis: {
        type: 'value'
    },
    yAxis: {
        type: 'category',
        data: xDataArr
    },
    series: [
        {
            type: 'bar',
            data: yDataArr
        }
    ]
}
```



3.1.3. 柱状图特点

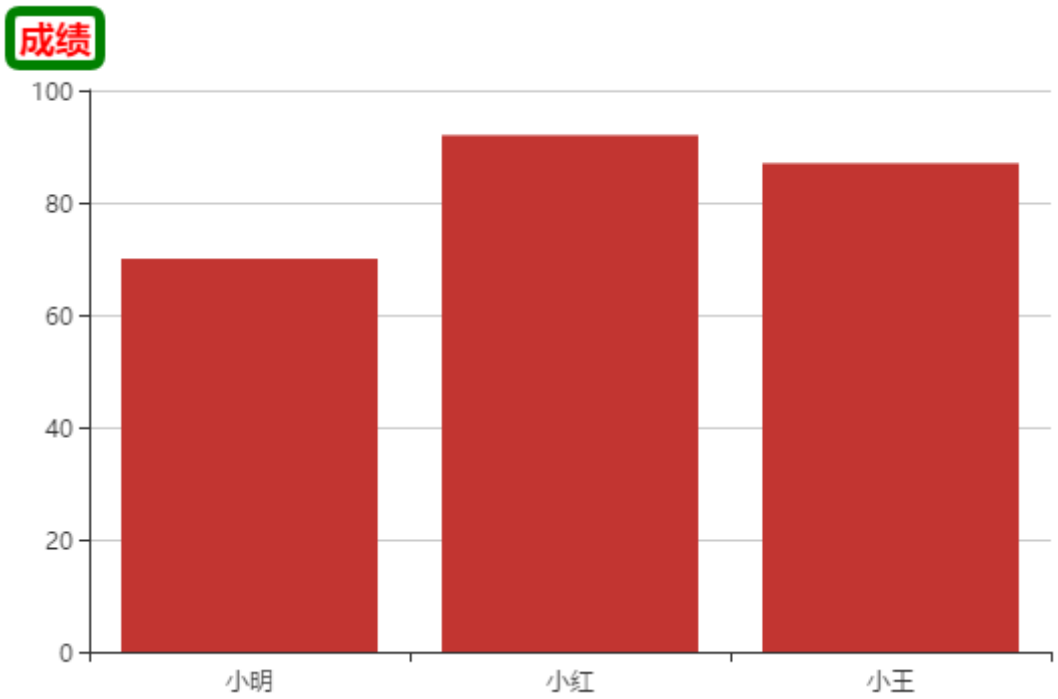
柱状图描述的是分类数据，呈现的是每一个分类中『有多少？』, 图表所表达出来的含义在于不同类别数据的排名\对比情况

3.1.4.通用配置

使用 `ECharts` 绘制出来的图表, 都天生就自带一些功能, 这些功能是每一个图表都具备的, 我们可以通过配置, 对这些功能进行设置.

- 标题: `title`

```
var option = {
    title: {
        text: "成绩",   // 标题文字
        textStyle: {
            color: 'red' // 文字颜色
        },
        borderWidth: 5,  // 标题边框
        borderColor: 'green', // 标题边框颜色
        borderRadius: 5,  // 标题边框圆角
        left: 20, // 标题的位置
        top: 20 // 标题的位置
    }
}
```



- 提示框: `tooltip`

  `tooltip`指的是当鼠标移入到图表或者点击图表时, 展示出的提示框

  - 触发类型: trigger

    可选值有`item\axis`

  - 触发时机: `triggerOn`

    可选值有 `mouseOver\click`

  - 格式化显示: `formatter`

    - 字符串模板

```
var option = {
    tooltip: {
        trigger: 'item',
        triggerOn: 'click',
        formatter: '{b}:{c}'
    }
}
```

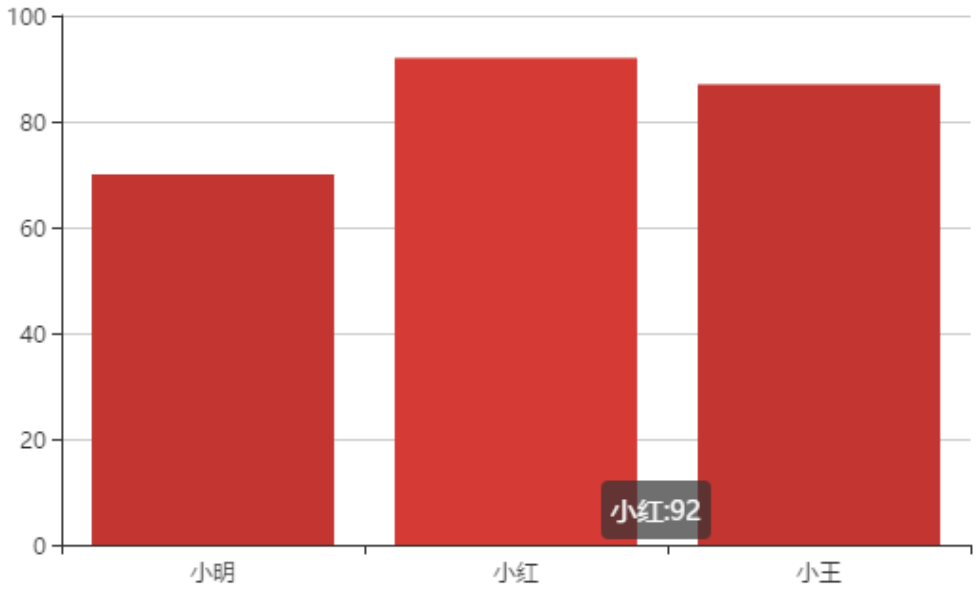
这个{b} 和 {c} 所代表的含义不需要去记，在官方文档中有详细的描述

模板变量有 {a}，{b}，{c}，{d}，{e}，分别表示系列名，数据名，数据值等。在 trigger 为 'axis' 的时候，会有多个系列的数据，此时可以通过 {a0}，{a1}，{a2} 这种后面加索引的方式表示系列的索引。不同图表类型下的 {a}，{b}，{c}，{d} 含义不一样。其中变量 {a}，{b}，{c}，{d} 在不同图表类型下代表数据含义为：

- 折线（区域）图、柱状（条形）图、K线图：{a}（系列名称），{b}（类目值），{c}（数值），{d}（无）
- 散点图（气泡）图：{a}（系列名称），{b}（数据名称），{c}（数值数组），{d}（无）
- 地图：{a}（系列名称），{b}（区域名称），{c}（合并数值），{d}（无）
- 饼图、仪表盘、漏斗图：{a}（系列名称），{b}（数据项名称），{c}（数值），{d}（百分比）

- 回调函数

```
var option = {
    tooltip: {
        trigger: 'item',
        triggerOn: 'click',
        formatter: function (arg) {
            return arg.name + ':' + arg.data
        }
    }
}
```

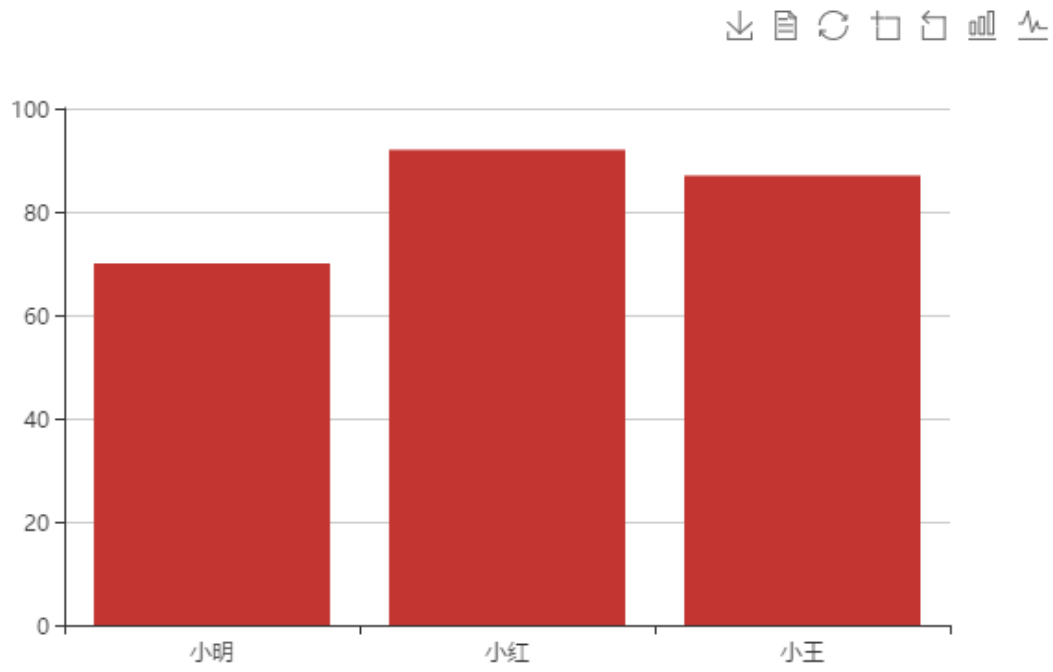


- 工具按钮: `toolbox`

  `toolbox`是`ECharts`提供的工具栏, 内置有 导出图片，数据视图, 重置, 数据区域缩放, 动态类型切换五个工具

  工具栏的按钮是配置在`feature`的节点之下

```
var option = {
    toolbox: {
        feature: {
            saveAsImage: {}, // 将图表保存为图片
            dataView: {}, // 是否显示出原始数据
            restore: {}, // 还原图表
            dataZoom: {}, // 数据缩放
            magicType: { // 将图表在不同类型之间切换,图表的转换需要数据的支持
                type: ['bar', 'line']
            }
        }
    }
}
```



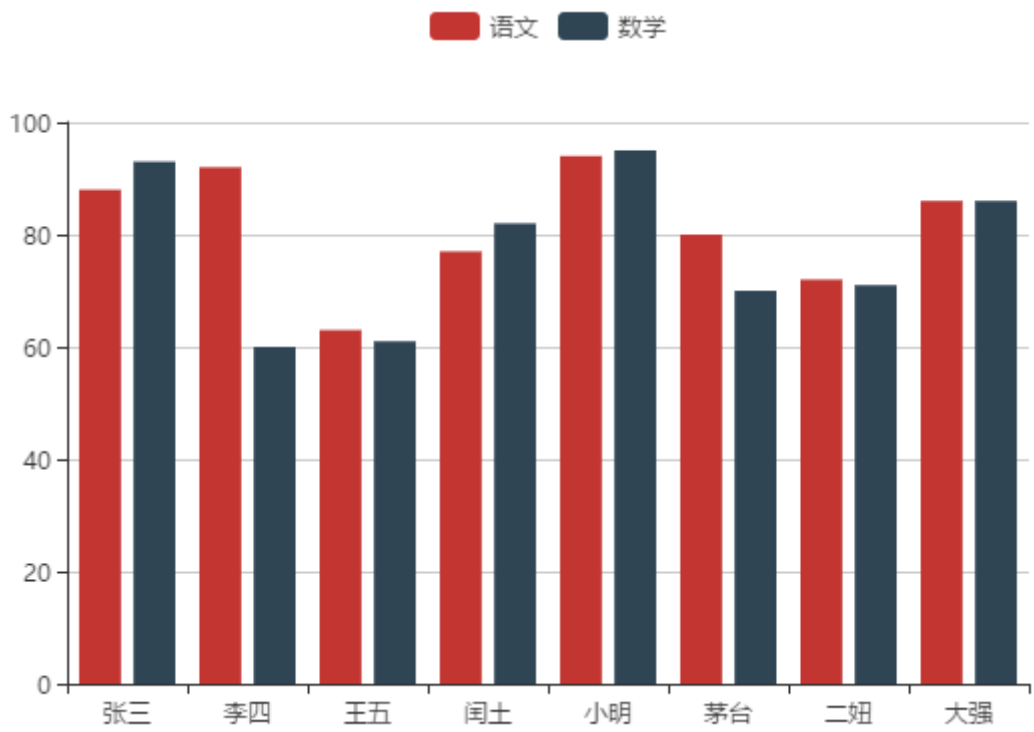
- 图例:`legend`

  `legend`是图例,用于筛选类别,需要和`series`配合使用

  - `legend`中的`data`是一个数组
  - `legend`中的`data`的值需要和`series`数组中某组数据的`name`值一致

```
var option = {
    legend: {
        data: ['语文', '数学']
    },
    xAxis: {
        type: 'category',
        data: ['张三', '李四', '王五', '闰土', '小明', '茅台', '二妞', '大强']
    },
    yAxis: {
        type: 'value'
    },
    series: [
        {
            name: '语文',
            type: 'bar',
            data: [88, 92, 63, 77, 94, 80, 72, 86]
        }, {
```

```
            name: '数学',
            type: 'bar',
            data: [93, 60, 61, 82, 95, 70, 71, 86]
        }
    ]
}
```



3.2.图表2 折线图

3.2.1.折线图的实现步骤

- 步骤1 `ECharts`最基本的代码结构

```
<!DOCTYPE html>
<html lang="en">
<head>
    <script src="js/echarts.min.js"></script>
</head>
<body>
    <div style="width: 600px;height:400px"></div>
    <script>
        var mCharts = echarts.init(document.querySelector("div"))
        var option = {}
        mCharts.setOption(option)
    </script>
</body>
</html>
```

此时`option`是一个空空如也的对象

- 步骤2 准备`x`轴的数据

```
var xDataArr = ['1月', '2月', '3月', '4月', '5月', '6月', '7月', '8月', '9月',
'10月', '11月', '12月']
```

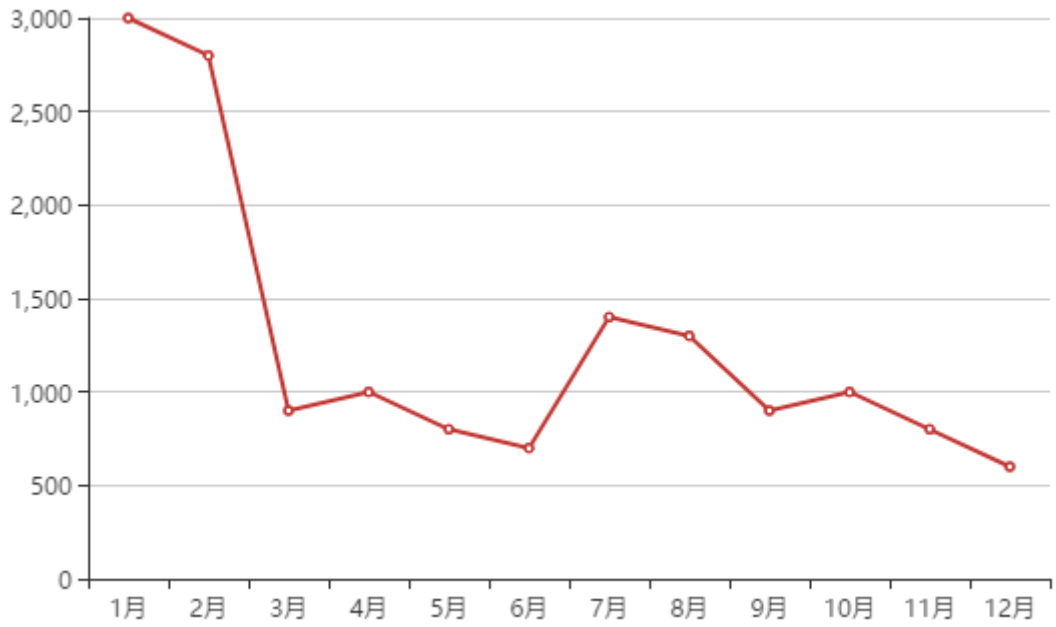
- 步骤3 准备`y`轴的数据

```
var yDataArr = [3000, 2800, 900, 1000, 800, 700, 1400, 1300, 900, 1000, 800, 600]
```

- 步骤4 准备`option`，将`series`中的`type`的值设置为: `line`

```
var option = {
    xAxis: {
        type: 'category',
        data: xDataArr
    },
    yAxis: {
        type: 'value'
    },
    series: [
        {
            type: 'line',
            data: yDataArr
        }
    ]
}
```

最终的效果如下:

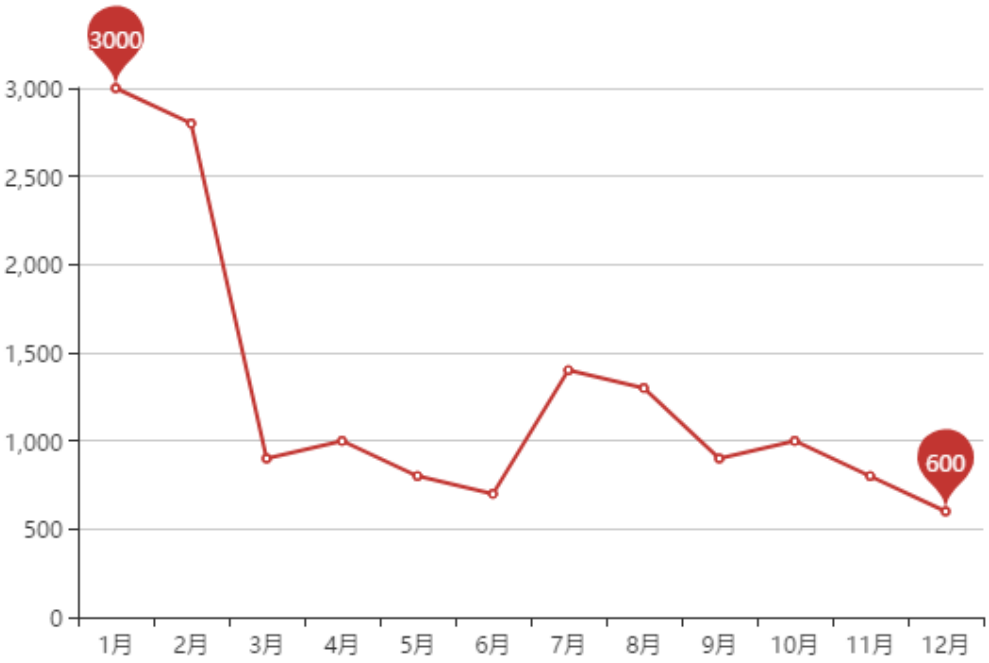


3.2.2.折线图的常见效果

- 标记
  - 最大值\最小值 `markPoint`

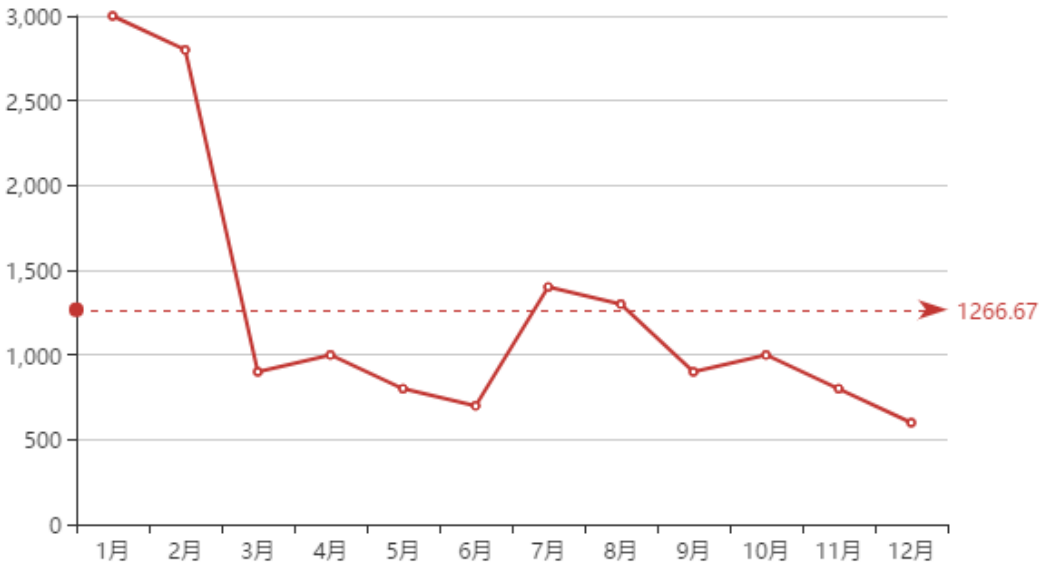
```
var option = {
    series: [
        {
            ......
            markPoint: {
```

```
                data: [
                    {
                        type: 'max',
                        name: '最大值'
                    }, {
                        type: 'min',
                        name: '最小值'
                    }
                ]
            }
        }
    ]
}
```



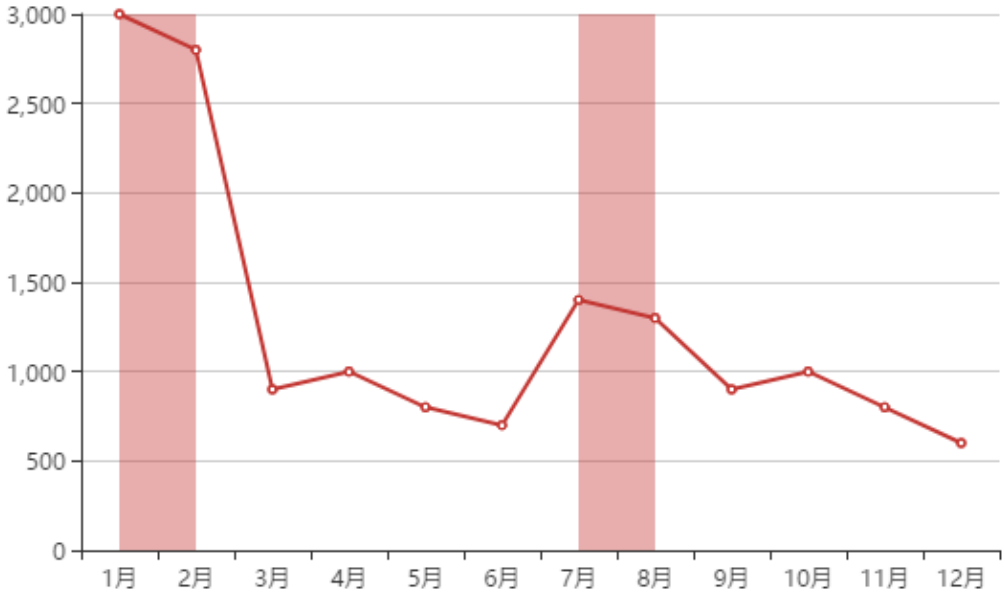
- 平均值 `markLine`

```
var option = {
    series: [
        {
            ......
            markLine: {
                data: [
                    {
                        type: 'average',
                        name: '平均值'
                    }
                ]
            }
        }
    ]
}
```



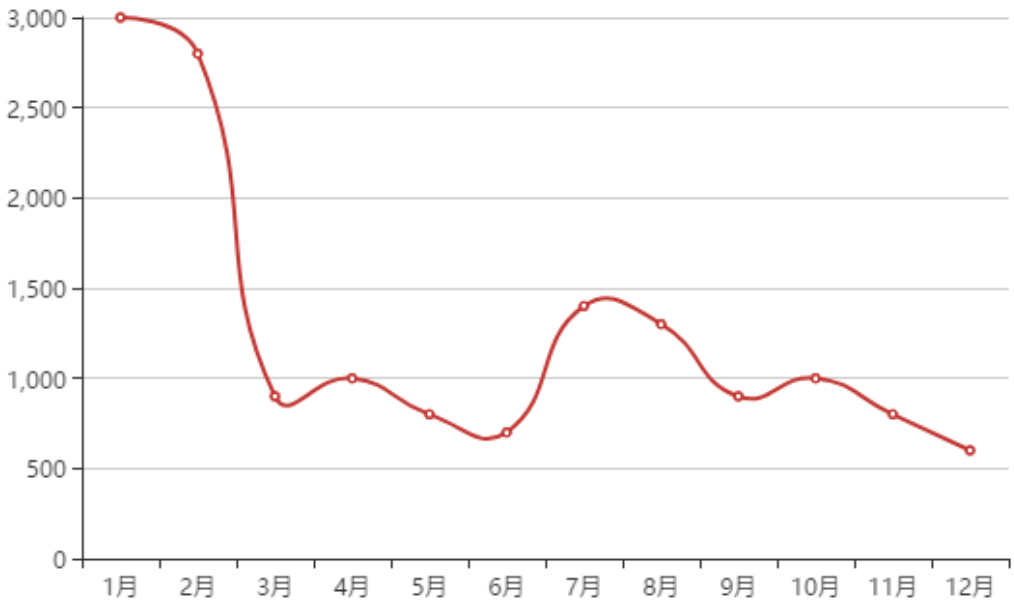
- 标注区间 `markArea`

```
var option = {
    series: [
        {
            ......
            markArea: {
              data: [
                [
                  {
                    xAxis: '1月'
                  }, {
                    xAxis: '2月'
                  }
                ],
                [
                  {
                    xAxis: '7月'
                  }, {
                    xAxis: '8月'
                  }
                ]
              ]
            }
        }
    ]
}
```



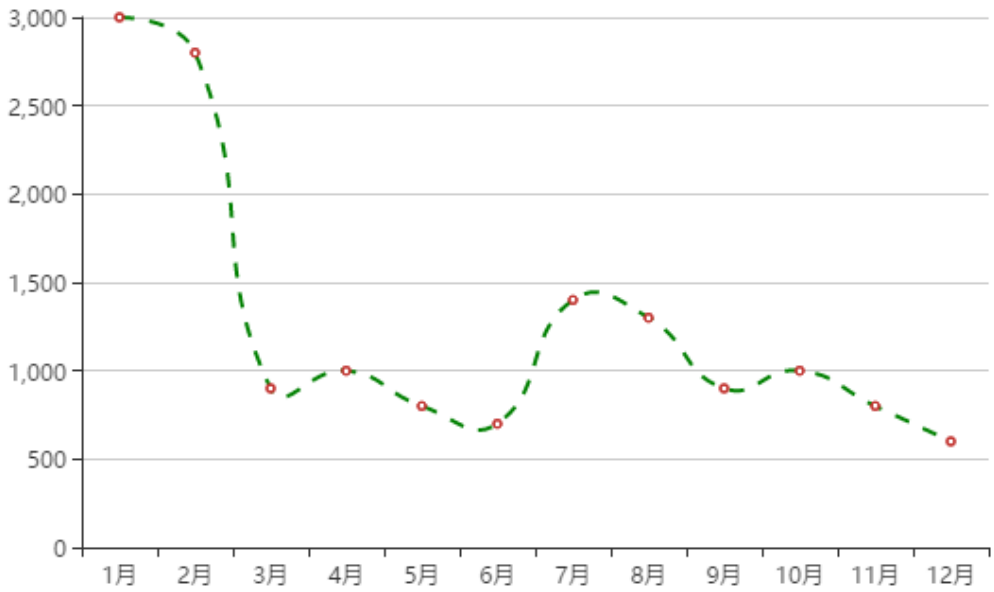
- 线条控制
  - 平滑线条 `smooth`

```
var option = {
    series: [
        {
            ......
            smooth: true
        }
    ]
}
```



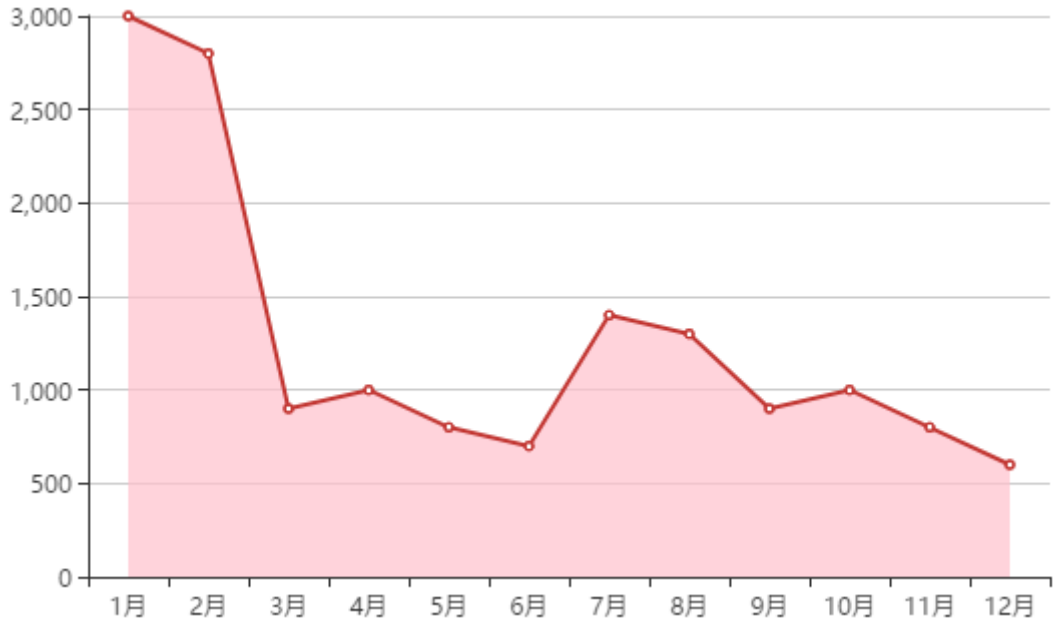
  - 线条样式 `lineStyle`

```
var option = {
    series: [
        {
            ......
            smooth: true,
            lineStyle: {
                color: 'green',
                type: 'dashed' // 可选值还有  dotted    solid
            }
        }
    ]
}
```



- 填充风格 `areaStyle`

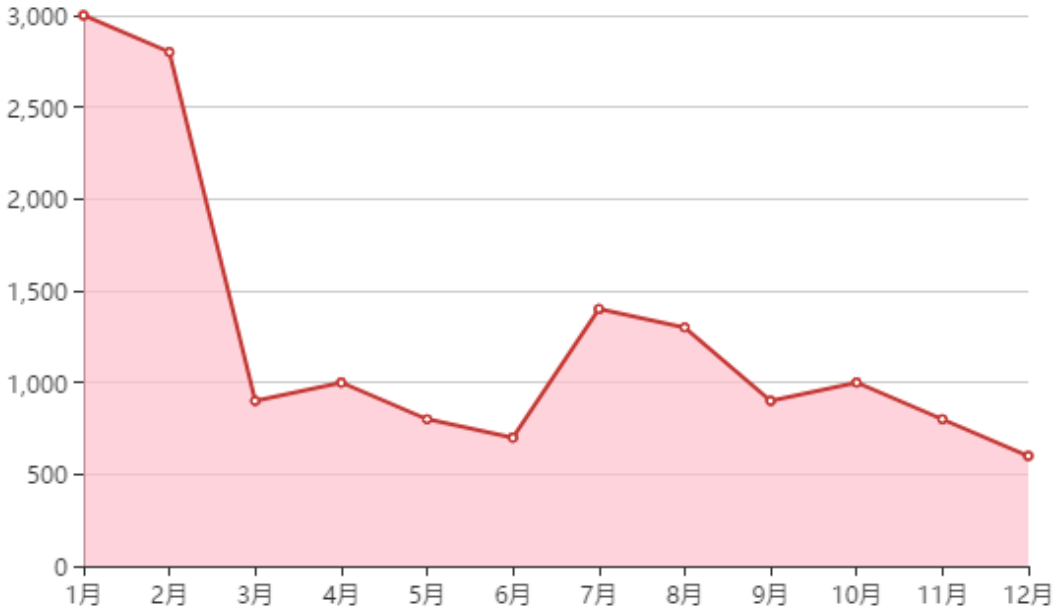
```
var option = {
    series: [
        {
            type: 'line',
            data: yDataArr,
            areaStyle: {
                color: 'pink'
            }
        }
    ]
}
```



- 紧挨边缘 `boundaryGap`

  `boundaryGap` 是设置给 `x` 轴的, 让起点从 `x` 轴的0坐标开始

```
var option = {
    xAxis: {
        type: 'category',
        data: xDataArr,
        boundaryGap: false
    }
}
```

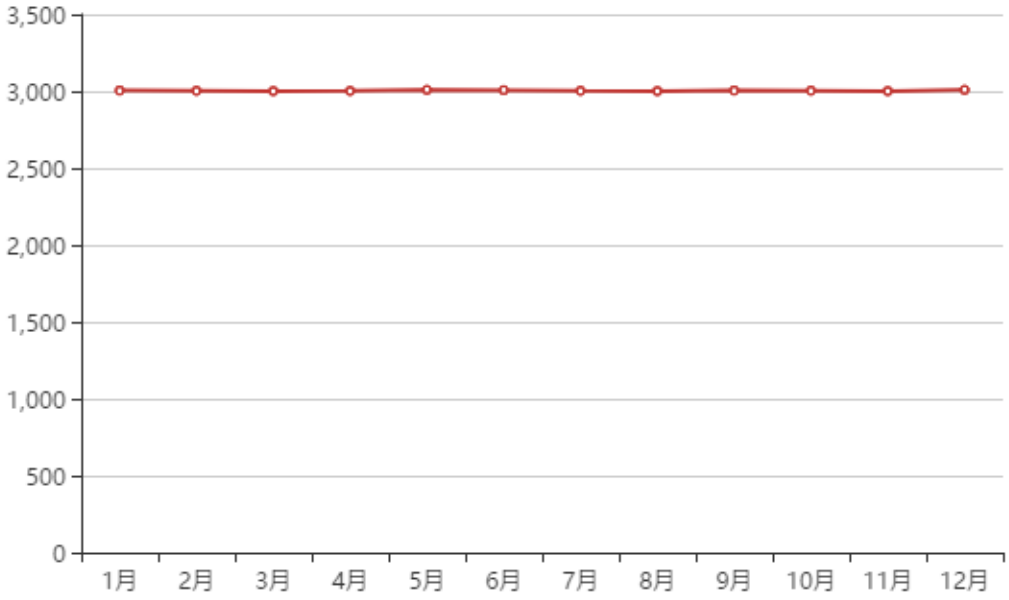


- 缩放, 脱离0值比例
  - 如果每一组数据之间相差较少, 且都比0大很多, 那么有可能会出现这种情况

```
var yDataArr = [3005, 3003, 3001, 3002, 3009, 3007, 3003, 3001, 3005,
3004, 3001, 3009]  // 此时y轴的数据都在3000附近, 每个数之间相差不多
var option = {
    xAxis: {
```

```
        type: 'category',
        data: xDataArr
    },
    yAxis: {
        type: 'value'
    },
    series: [
        {
            type: 'line',
            data: yDataArr
        }
    ]
}
```

效果如下图:

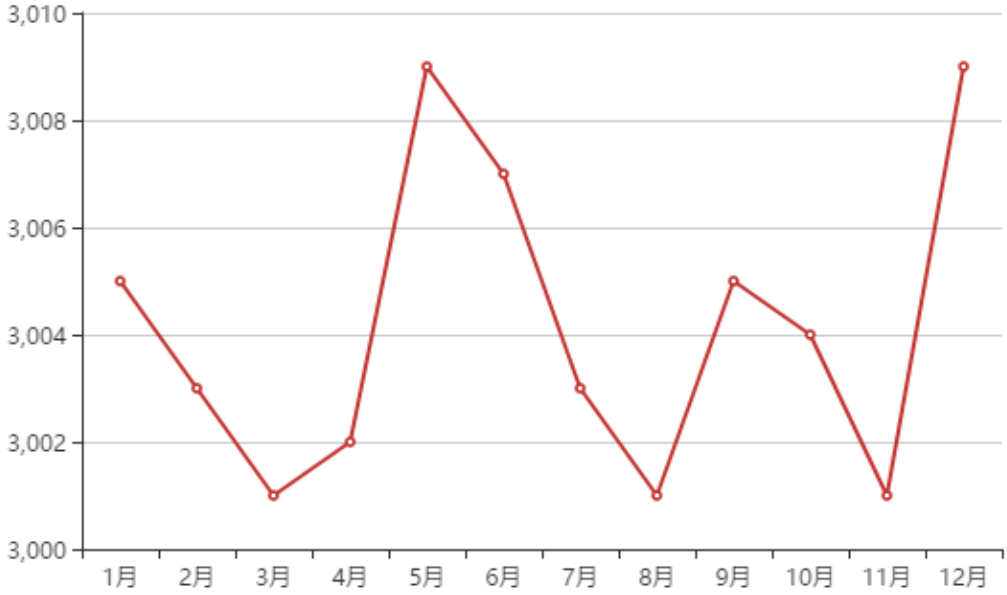


这显然不是我们想要的效果, 因此可以配置上`scale`, 让其摆脱0值比例

- `scale`配置

  `scale`应该配置给`y`轴

```
var option = {
    yAxis: {
        type: 'value',
        scale: true
    }
}
```

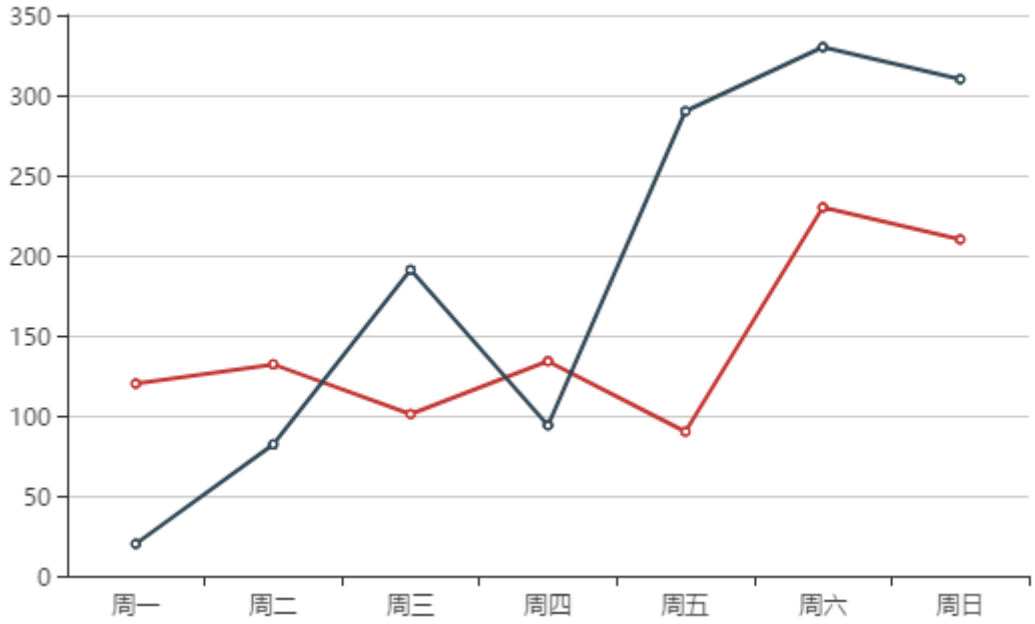


- 堆叠图

  堆叠图指的是, 同个类目轴上系列配置相同的`stack`值后，后一个系列的值会在前一个系列的值上相加

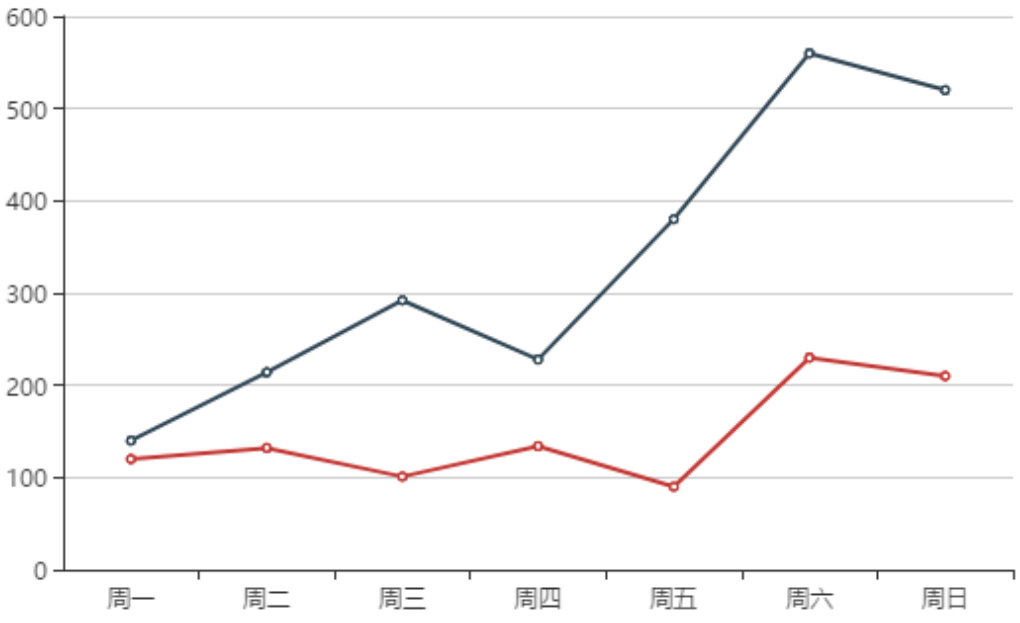
  如果在一个图表中有两个或者多个折线图, 在没有使用堆叠配置的时候, 效果如下:

```
<script>
  var mCharts = echarts.init(document.querySelector("div"))
  var xDataArr =  ['周一', '周二', '周三', '周四', '周五', '周六', '周日']
  var yDataArr1 = [120, 132, 101, 134, 90, 230, 210]
  var yDataArr2 = [20, 82, 191, 94, 290, 330, 310]
  var option = {
    xAxis: {
      type: 'category',
      data: xDataArr
    },
    yAxis: {
      type: 'value',
      scale: true
    },
    series: [
      {
        type: 'line',
        data: yDataArr1
      },
      {
        type: 'line',
        data: yDataArr2
      }
    ]
  }
  mCharts.setOption(option)
</script>
```



使用了堆叠图之后:

```
var option = {
    series: [
        {
            type: 'line',
            data: yDataArr1,
            stack: 'all'  // series中的每一个对象配置相同的stack值，这个all可以任意写
        },
        {
            type: 'line',
            data: yDataArr2,
            stack: 'all' // series中的每一个对象配置相同的stack值，这个all可以任意写
        }
    ]
}
```



蓝色这条线的y轴起点, 不再是y轴, 而是红色这条线对应的点. 所以相当于蓝色是在红色这条线的基础之上进行绘制. 基于前一个图表进行堆叠

3.2.3.折线图的特点

折线图更多的使用来呈现数据随时间的『变化趋势』

3.3.图表3 散点图

3.3.1.散点图的实现步骤

- 步骤1 `ECharts`最基本的代码结构

```
<!DOCTYPE html>
<html lang="en">
<head>
    <script src="js/echarts.min.js"></script>
</head>
<body>
    <div style="width: 600px;height:400px"></div>
    <script>
        var mCharts = echarts.init(document.querySelector("div"))
        var option = {}
        mCharts.setOption(option)
    </script>
</body>
</html>
```

此时`option`是一个空空如也的对象

- 步骤2 准备`x`轴和`y`轴的数据

```
var data = [{ "gender": "female", "height": 161.2, "weight": 51.6 }, {
"gender": "female", "height": 167.5, "weight": 59 }, { "gender": "female",
"height": 159.5, "weight": 49.2 }, { "gender": "female", "height": 157,
"weight": 63 }, { "gender": "female", "height": 155.8, "weight": 53.6 }, {
"gender": "female", "height": 170, "weight": 59 }, { "gender": "female",
"height": 159.1, "weight": 47.6 }, { "gender": "female", "height": 166,
"weight": 69.8 }, { "gender": "female", "height": 176.2, "weight": 66.8 }, {
"gender": "female", "height": 160.2, "weight": 75.2 }, { "gender": "female",
"height": 172.5, "weight": 55.2 }, { "gender": "female", "height": 170.9,
"weight": 54.2 }, { "gender": "female", "height": 172.9, "weight": 62.5 }, {
"gender": "female", "height": 153.4, "weight": 42 }, { "gender": "female",
"height": 160, "weight": 50 }, { "gender": "female", "height": 147.2,
"weight": 49.8 },...此处省略...]
```

假设这个数据是从服务器获取到的, 数组中的每一个元素都包含3个维度的数据: 性别,身高,体重, 而散点图需要的数据是一个二维数组, 所以我们需要将从服务器获取到的这部分数据,通过代码生成散点图需要的数据

```
var axisData = []
for (var i = 0; i < data.length; i++) {
    var height = data[i].height
    var weight = data[i].weight
    var itemArr = [height, weight]
    axisData.push(itemArr)
}
```

`axisData`就是一个二维数组, 数组中的每一个元素还是一个数组, 最内层数组中有两个元素, 一个代表身高, 一个代表体重

- 步骤3 准备配置项
  - `xAxis`和`yAxis`的`type`都要设置为`value`
  - 在`series`下设置`type:scatter`

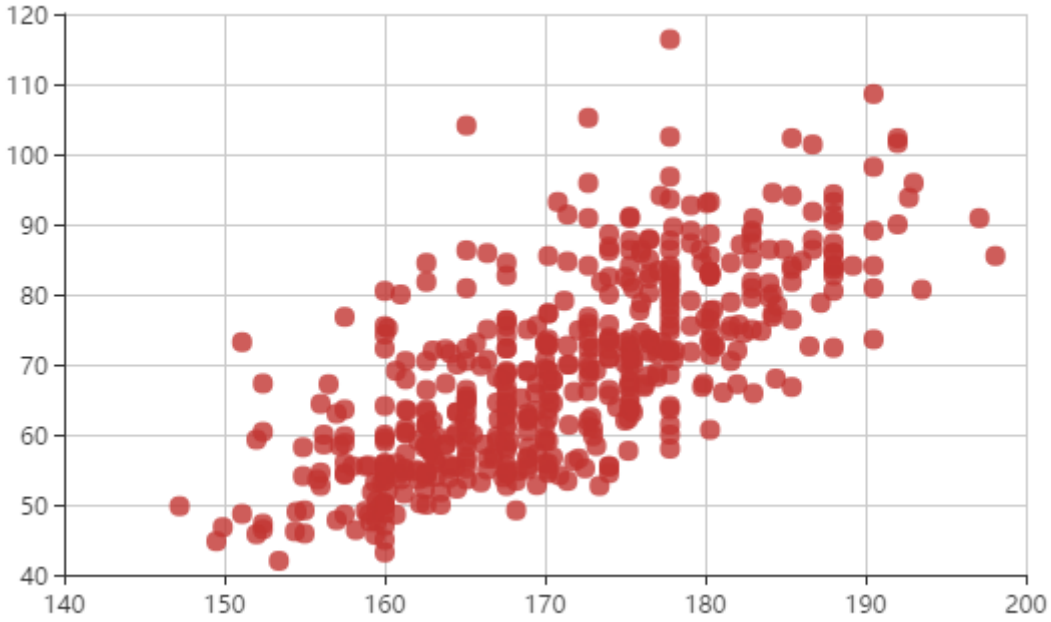
```
var option = {
    xAxis: {
        type: 'value'
    },
    yAxis: {
        type: 'value'
    },
    series: [
        {
            type: 'scatter',
            data: axisData
        }
    ]
}
```

- 步骤4 调整配置项, 脱离0值比例

  给`xAxis`和`yAxis`配置`scale`的值为`true`

```
var option = {
    xAxis: {
        type: 'value',
        scale: true
    },
    yAxis: {
        type: 'value',
        scale: true
    },
    series: [
        {
            type: 'scatter',
            data: axisData,

        }
    ]
}
```



3.3.2.散点图的常见效果

- 气泡图效果

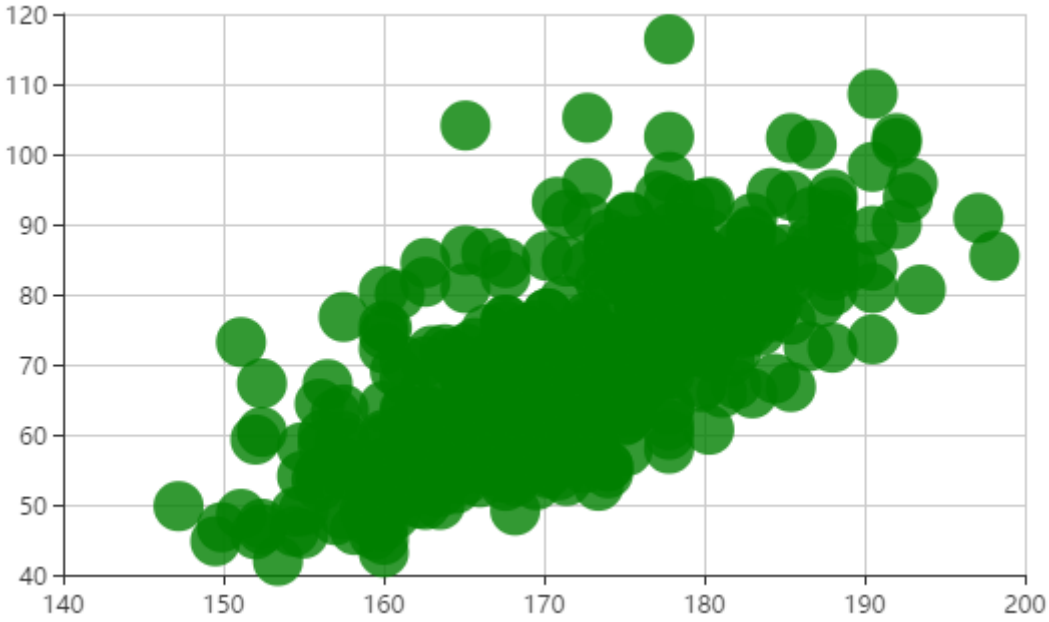
  要能够达到气泡图的效果, 其实就是让每一个散点的大小不同, 让每一个散点的颜色不同

  - `symbolSize` 控制散点的大小
  - `itemStyle.color` 控制散点的颜色

  这两个配置项都支持固定值的写法, 也支持回调函数的写法

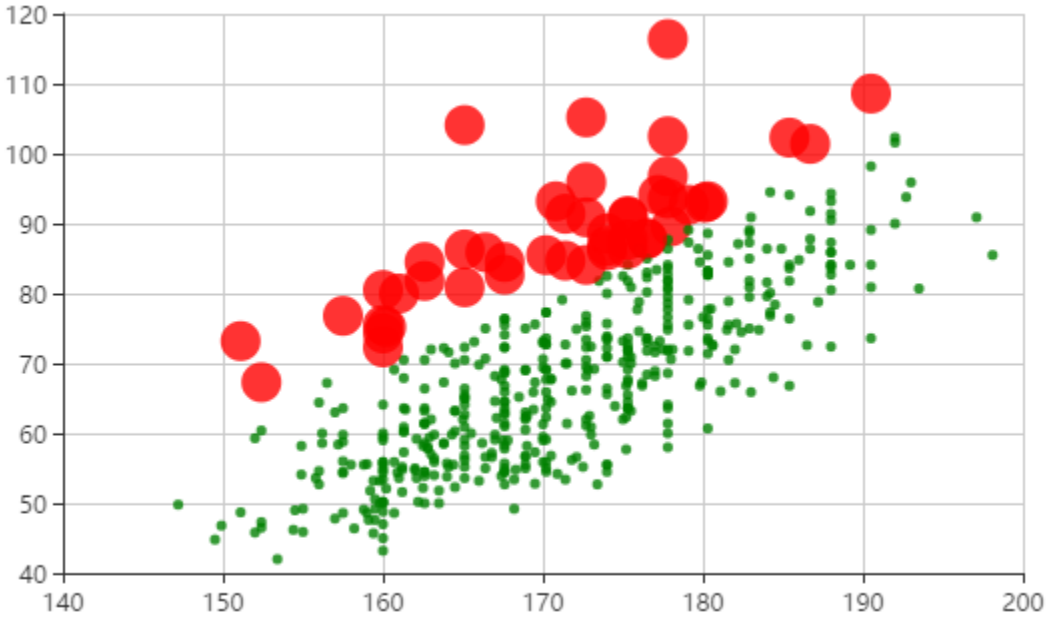
  固定值的写法如下:

```
var option = {
    series: [
        {
            type: 'scatter',
            data: axisData,
            symbolSize: 25,
            itemStyle: {
                color: 'green',
            }
        }
    ]
}
```



回调函数的写法如下:

```
var option = {
    series: [
        {
            type: 'scatter',
            data: axisData,
            symbolSize: function (arg) {
                var weight = arg[1]
                var height = arg[0] / 100
                // BMI > 28 则代表肥胖，肥胖的人用大的散点标识，正常的人用小散点标识
                // BMI: 体重/ 身高*身高     kg  m
                var bmi = weight / (height * height)
                if (bmi > 28) {
                    return 20
                }
                return 5
            },
            itemStyle: {
                color: function (arg) {
                    var weight = arg.data[1]
                    var height = arg.data[0] / 100
                    var bmi = weight / (height * height)
                    if (bmi > 28) {
                        return 'red'
                    }
                    return 'green'
                }
            }
        }
    ]
}
```

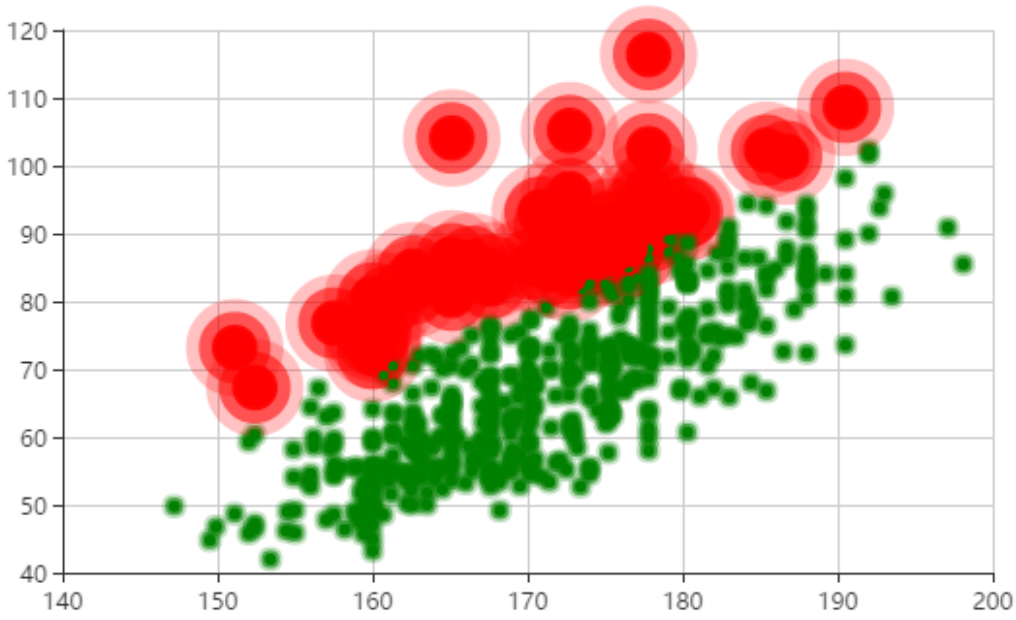


- 涟漪动画效果
  - `type:effectScatter`

    将`type`的值从`scatter`设置为`effectScatter`就能够产生涟漪动画的效果
  - `rippleEffect`

    `rippleEffect`可以配置涟漪动画的大小

```
var option = {
    series: [
        {
            type: 'effectScatter',
            rippleEffect:{
                scale:3
            }
        }
    ]
}
```



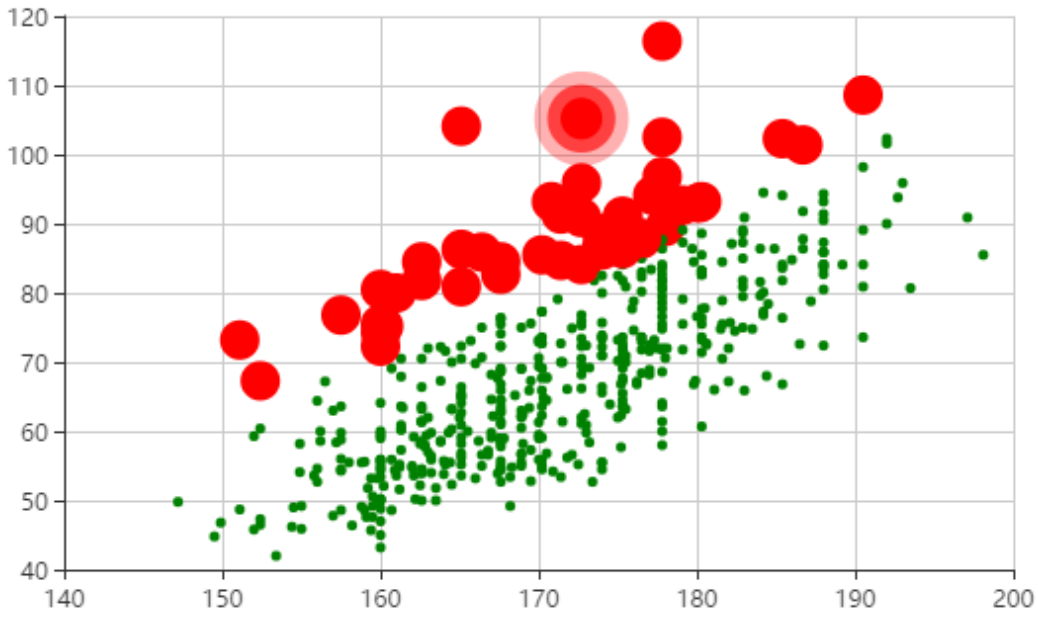
- `showEffectOn`

  `showEffectOn`可以控制涟漪动画在什么时候产生, 它的可选值有两个: `render`和`emphasis`

  `render`代表界面渲染完成就开始涟漪动画

  `emphasis`[强 调 重 视 重 要 性'emfəsɪ s]鼠标移过某个散点的时候, 该散点开始涟漪动画

```
var option = {
    series: [
        {
            type: 'effectScatter',
            showEffectOn: 'emphasis',
            rippleEffect:{
                scale:3
            }
        }
    ]
}
```



- 结合地图

  散点图也经常结合地图来进行地图区域的标注, 这个效果将在讲解地图时实现

3.3.3.散点图的特点

散点图可以帮助我们推断出不同维度数据之间的相关性, 比如上述例子中,看得出身高和体重是正相关, 身高越高, 体重越重

散点图也经常用在地图的标注上

3.3.4.直角坐标系的常见配置

直角坐标系的图表指的是带有x轴和y轴的图表, 常见的直角坐标系的图表有: 柱状图 折线图 散点图

针对于直角坐标系的图表, 有一些通用的配置

- 配置1: 网格 `grid`

  grid是用来控制直角坐标系的布局和大小, x轴和y轴就是在grid的基础上进行绘制的

- 显示`grid`

  `show: true`

- `grid`的边框

  `borderWidth`: 10

- `grid`的位置和大小

  `left` `top` `right` `bottom`

  `width` `height`

```
var option = {
    grid: {
        show: true, // 显示grid
        borderWidth: 10, // grid的边框宽度
        borderColor: 'red', // grid的边框颜色
        left: 100, // grid的位置
        top: 100,
        width: 300, // grid的大小
        height: 150
    }
}
```

- 配置2: 坐标轴 `axis`

  坐标轴分为x轴和y轴, 一个`grid`中最多有两种位置的`x`轴和`y`轴

  - 坐标轴类型 `type`

    `value`: 数值轴, 自动会从目标数据中读取数据

    `category`: 类目轴, 该类型必须通过`data`设置类目数据

  - 坐标轴位置

    `xAxis`: 可取值为 `top`或者`bottom`

    `yAxis`: 可取值为`left`或者`right`

```
var option = {
    xAxis: {
        type: 'category',
        data: xDataArr,
        position: 'top'
    },
    yAxis: {
        type: 'value',
        position: 'right'
    }
}
```

- 配置3: 区域缩放 `dataZoom`

  `dataZoom` 用于区域缩放, 对数据范围过滤, x轴和y轴都可以拥有, `dataZoom` 是一个数组, 意味着可以配置多个区域缩放器

  - 区域缩放类型 `type`

    `slider`: 滑块

    `inside`: 内置, 依靠鼠标滚轮或者双指缩放

- 产生作用的轴

  `xAxisIndex`:设置缩放组件控制的是哪个`x`轴, 一般写0即可

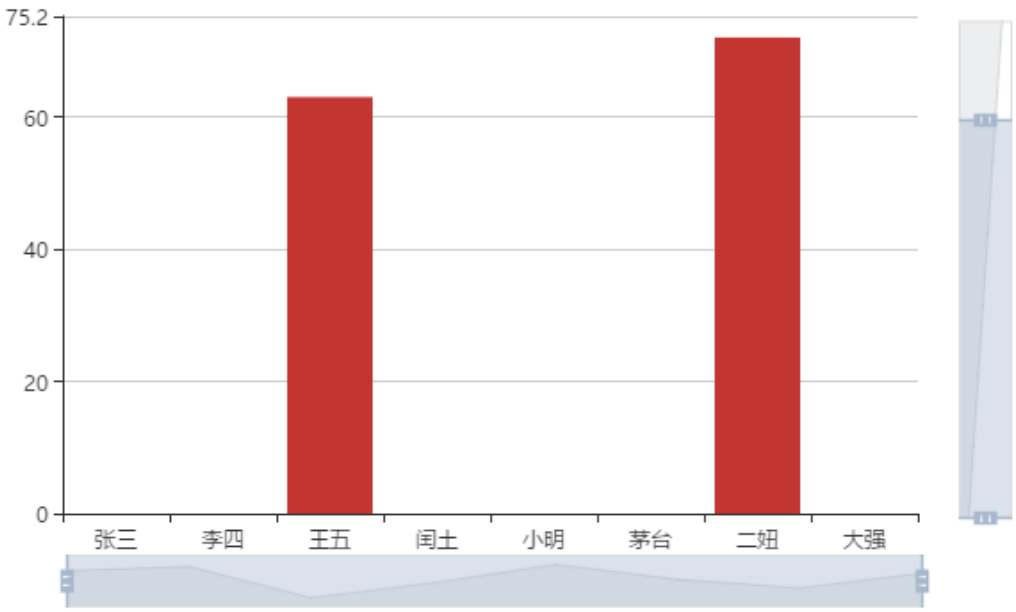
  `yAxisIndex`:设置缩放组件控制的是哪个`y`轴, 一般写0即可

- 指明初始状态的缩放情况

  `start`: 数据窗口范围的起始百分比

  `end`: 数据窗口范围的结束百分比

```
var option = {
    xAxis: {
        type: 'category',
        data: xDataArr
    },
    yAxis: {
        type: 'value'
    },
    dataZoom: [
        {
            type: 'slider',
            xAxisIndex: 0
        },
        {
            type: 'slider',
            yAxisIndex: 0,
            start: 0,
            end: 80
        }
    ]
}
```



需要注意的是, 针对于非直角坐标系图表, 比如饼图 地图 等, 以上三个配置可能就不会生效了

3.4.图表4 饼图

3.4.1.饼图的实现步骤

- 步骤1 `ECharts`最基本的代码结构

```
<!DOCTYPE html>
<html lang="en">
<head>
    <script src="js/echarts.min.js"></script>
</head>
<body>
    <div style="width: 600px;height:400px"></div>
    <script>
        var mCharts = echarts.init(document.querySelector("div"))
        var option = {}
        mCharts.setOption(option)
    </script>
</body>
</html>
```

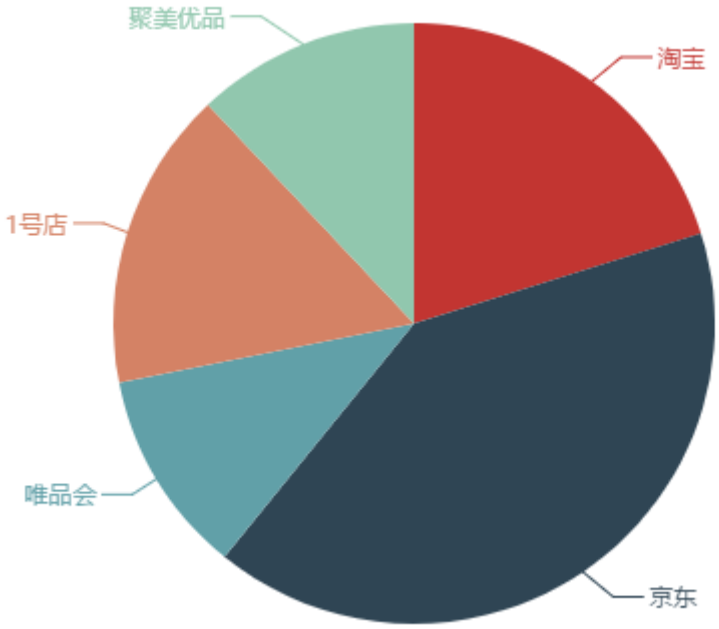
此时`option`是一个空空如也的对象

- 步骤2 准备数据

```
var pieData = [
    {
        value: 11231,
        name: "淘宝",
    },
    {
        value: 22673,
        name: "京东"
    },
    {
        value: 6123,
        name: "唯品会"
    },
    {
        value: 8989,
        name: "1号店"
    },
    {
        value: 6700,
        name: "聚美优品"
    }
]
```

- 步骤3 准备配置项 在`series`下设置 `type:pie`

```
var option = {
    series: [
        {
            type: 'pie',
            data: pieData
        }
    ]
}
```



注意:

- 饼图的数据是由`name`和`value`组成的字典所形成的数组
- 饼图无须配置`xAxis`和`yAxis`

3.4.2.饼图的常见效果

- 显示数值
  - `label.show`: 显示文字
  - `label.formatter`: 格式化文字

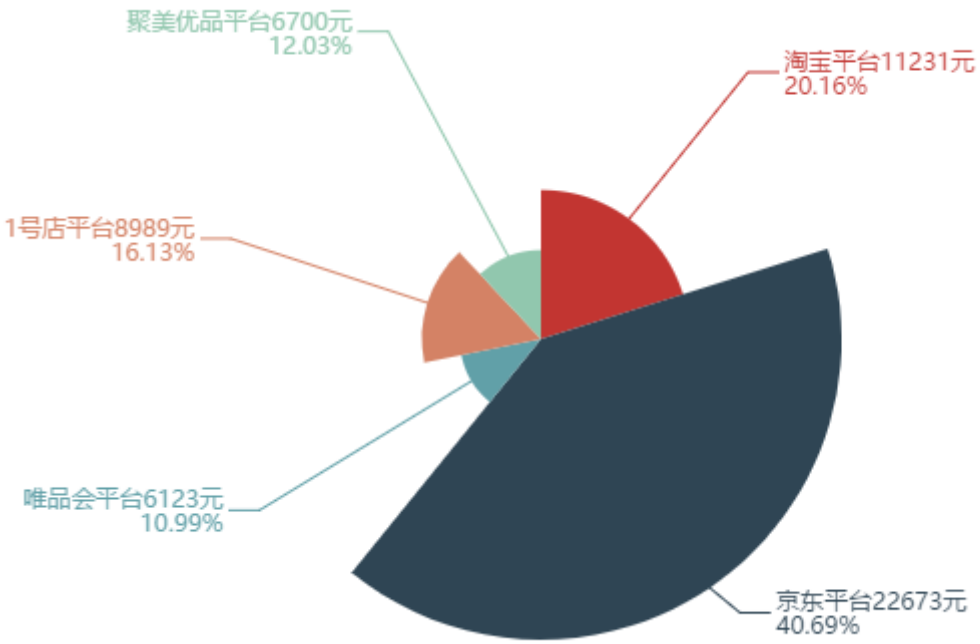
```
var option = {
    series: [
      {
        type: 'pie',
        data: pieData,
        label: {
          show: true,
          formatter: function (arg) {
            return arg.data.name + '平台' + arg.data.value + '元\n' +
arg.percent + '%'
          }
        }
      }
    ]
  }
```

- 南丁格尔图

  南丁格尔图指的是每一个扇形的半径随着数据的大小而不同, 数值占比越大, 扇形的半径也就越大
  - `roseType:'radius'`

```
  var option = {
    series: [
```

```
      {
        type: 'pie',
        data: pieData,
        label: {
          show: true,
          formatter: function (arg) {
            return arg.data.name + '平台' + arg.data.value + '元\n' +
arg.percent + '%'
          }
        },
        roseType: 'radius'
      }
    ]
  }
```

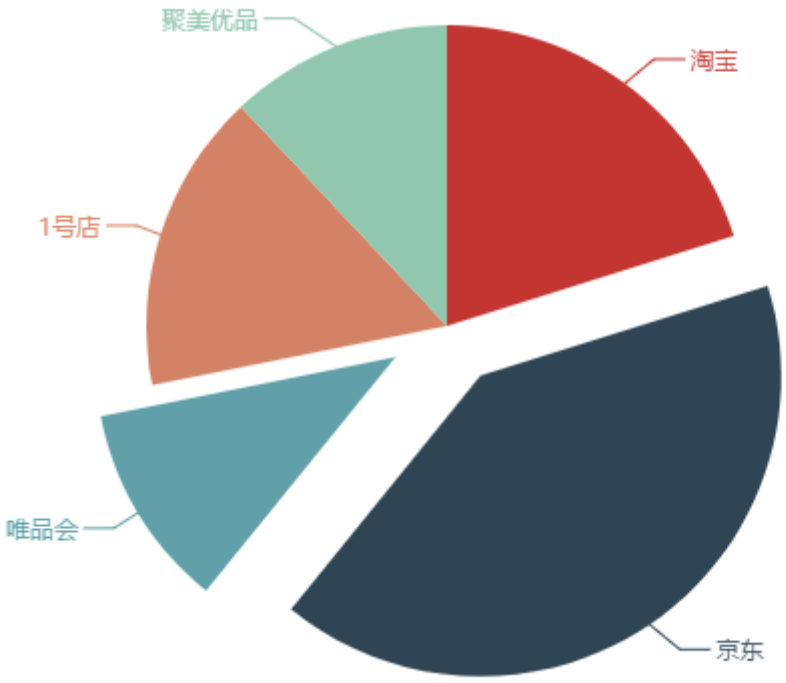


- 选中效果
  - `selectedMode: 'multiple'`

    选中模式，表示是否支持多个选中，默认关闭，支持布尔值和字符串，字符串取值可选`'single'`，`'multiple'`，分别表示单选还是多选
  - `selectedOffset: 30`

    选中扇区的偏移距离

```
  var option = {
    series: [
      {
        type: 'pie',
        data: pieData,
        selectedMode: 'multiple', //
        selectedOffset: 30
      }
    ]
  }
```

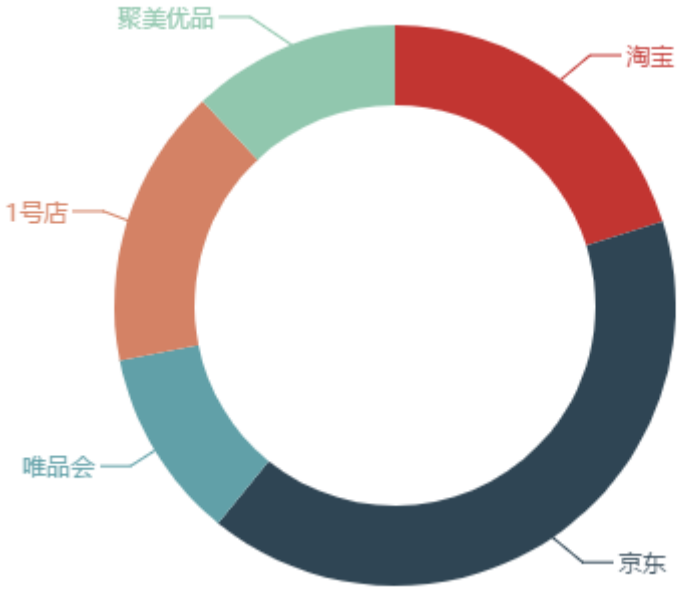


- 圆环
  - `radius`

  饼图的半径。可以为如下类型：

  `number`：直接指定外半径值。 `string`：例如，`'20%'`，表示外半径为可视区尺寸（容器高宽中较小一项）的 20% 长度。 `Array.`：数组的第一项是内半径，第二项是外半径, 通过`Array`, 可以将饼图设置为圆环图

```
var option = {
    series: [
        {
            type: 'pie',
            data: pieData,
            radius: ['50%', '70%']
        }
    ]
}
```



3.4.3.饼图的特点

饼图可以很好地帮助用户快速了解不同分类的数据的占比情况

3.5.图表5 地图

3.5.1.地图图表的使用方式

`百度地图API`: 使用百度地图的`api`, 它能够在线联网展示地图, 百度地图需要申请`ak`

`矢量地图`: 可以离线展示地图, 需要开发者准备矢量地图数据

接下来的实现是通过矢量图的方式来实现的

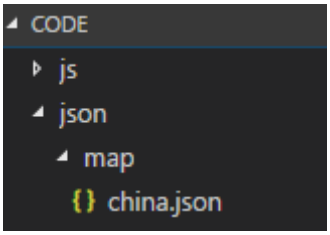
3.5.2.矢量地图的实现步骤

- 步骤1 `ECharts`最基本的代码结构

```
<!DOCTYPE html>
<html lang="en">
<head>
    <script src="js/echarts.min.js"></script>
</head>
<body>
    <div style="width: 600px;height:400px"></div>
    <script>
        var mCharts = echarts.init(document.querySelector("div"))
        var option = {}
        mCharts.setOption(option)
    </script>
</body>
</html>
```

  此时`option`是一个空空如也的对象

- 步骤2 准备中国的矢量`json`文件, 放到 `json/map/`目录之下



- 步骤3 使用`Ajax`获取`china.json`

```
$.get('json/map/china.json', function (chinaJson) {

})
```

- 步骤4 在Ajax的回调函数中, 往`echarts`全局对象注册地图的`json`数据

`echarts.registerMap('chinaMap', chinaJson)`

```
$.get('json/map/china.json', function (chinaJson) {
  echarts.registerMap('chinaMap', chinaJson)
})
```

- 步骤5 获取完数据之后, 需要配置`geo`节点, 再次的`setOption`

`type`: `'map'`

`map`: `'chinaMap'`

```
var mCharts = echarts.init(document.querySelector("div"))
$.get('json/map/china.json', function (chinaJson) {
    echarts.registerMap('chinaMap', chinaJson)
    var option = {
        geo: {
            type: 'map',// map是一个固定的值
            map: 'chinaMap',//chinaMap需要和registerMap中的第一个参数保持一致
        }
    };
    mCharts.setOption(option)
})
```

注意：需要注意的是, 由于在代码中使用了`Ajax`, 所以, 关于此文件的打开, 不能以`file`的协议打开, 应该将其置于`HTTP`的服务之下方可正常展示地图

3.5.3.地图的常见配置

- 缩放拖动：`roam`

```
var option = {
    geo: {
        type: 'map',// map是一个固定的值
        map: 'chinaMap',//chinaMap需要和registerMap中的第一个参数保持一致,
        roam: true, // 运行使用鼠标进行拖动和缩放
    }
}
```

- 名称显示：`label`

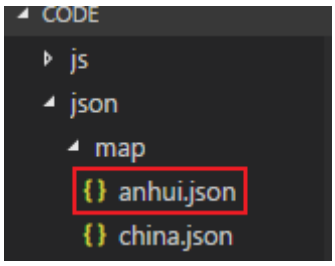
```
var option = {
    geo: {
        type: 'map',// map是一个固定的值
        map: 'chinaMap',//chinaMap需要和registerMap中的第一个参数保持一致,
        roam: true,
        label: {
            show: true
        }
    }
}
```

- 初始缩放比例：`zoom`
- 地图中心点：`center`

```
var option = {
    geo: {
        type: 'map',// map是一个固定的值
        map: 'chinaMap',//chinaMap需要和registerMap中的第一个参数保持一致,
        roam: true,
        label: {
            show: true
        },
        zoom: 0.8, // 地图的缩放比例，大于1代表放大，小于1代表缩小
        center: [87.617733, 43.792818] // 当前视角的中心点，用经纬度表示
    }
}
```

3.5.4.地图的常见效果

- 显示某个区域
  - 准备安徽省的矢量地图数据



  - 加载安徽省地图的矢量数据

```
$.get('json/map/anhui.json', function (anhuiJson) {

})
```

  - 在Ajax的回调函数中注册地图矢量数据

    `echarts.registerMap('anhui', anhuiJson)`
  - 配置`geo`的`type:'map'`,`map:'anhui'`
  - 通过`zoom`调整缩放比例
  - 通过`center`调整中心点

```
<script>
    var mCharts = echarts.init(document.querySelector("div"))
    $.get('json/map/anhui.json', function (anhuiJson) {
        console.log(anhuiJson)
        echarts.registerMap('anhui', anhuiJson)
        var option = {
            geo: {
```

```
                type: 'map',
                map: 'anhui',
                label: {
                    show: true
                },
                zoom: 1.2,
                center: [116.507676, 31.752889]
            }
        };
        mCharts.setOption(option)
    })
</script>
```

- 不同城市颜色不同
  - 1.显示基本的中国地图

```
<body>
    <div style="width: 600px;height:400px;border:1px solid red"></div>
    <script>
        var mCharts = echarts.init(document.querySelector("div"))
        $.get('json/map/china.json', function (chinaJson) {
            echarts.registerMap('chinaMap', chinaJson)
            var option = {
                geo: {
                    type: 'map',
                    map: 'chinaMap',
                    roam: true,
                    label: {
                        show: true
                    }
                }
            }
            mCharts.setOption(option)
        })
    </script>
</body>
</html>
```

- 2.准备好城市空气质量的数据, 并且将数据设置给`series`

```
var airData = [{ name: '北京', value: 39.92 },{ name: '天津', value:
39.13 },{ name: '上海', value: 31.22 },{ name: '重庆', value: 66 },{
name: '河北', value: 147 },{ name: '河南', value: 113 },{ name: '云南',
value: 25.04 },{ name: '辽宁', value: 50 },{ name: '黑龙江', value: 114
},{ name: '湖南', value: 175 },{ name: '安徽', value: 117 },{ name: '山
东', value: 92 },{ name: '新疆', value: 84 },{ name: '江苏', value: 67 },
{ name: '浙江', value: 84 },{ name: '江西', value: 96 },{ name: '湖北',
value: 273 },{ name: '广西', value: 59 },{ name: '甘肃', value: 99 },{
name: '山西', value: 39 },{ name: '内蒙古', value: 58 },{ name: '陕西',
value: 61 },{ name: '吉林', value: 51 },{ name: '福建', value: 29 },{
name: '贵州', value: 71 },{ name: '广东', value: 38 },{ name: '青海',
value: 57 },{ name: '西藏', value: 24 },{ name: '四川', value: 58 },{
name: '宁夏', value: 52 },{ name: '海南', value: 54 },{ name: '台湾',
value: 88 },{ name: '香港', value: 66 },{ name: '澳门', value: 77 },{
name: '南海诸岛', value: 55 }]
......
var option = {
    ......
    series: [
        {
            data: airData
        }
    ]
}
```

- 3.将`series`下的数据和`geo`关联起来

`geoIndex: 0`

`type: 'map'`

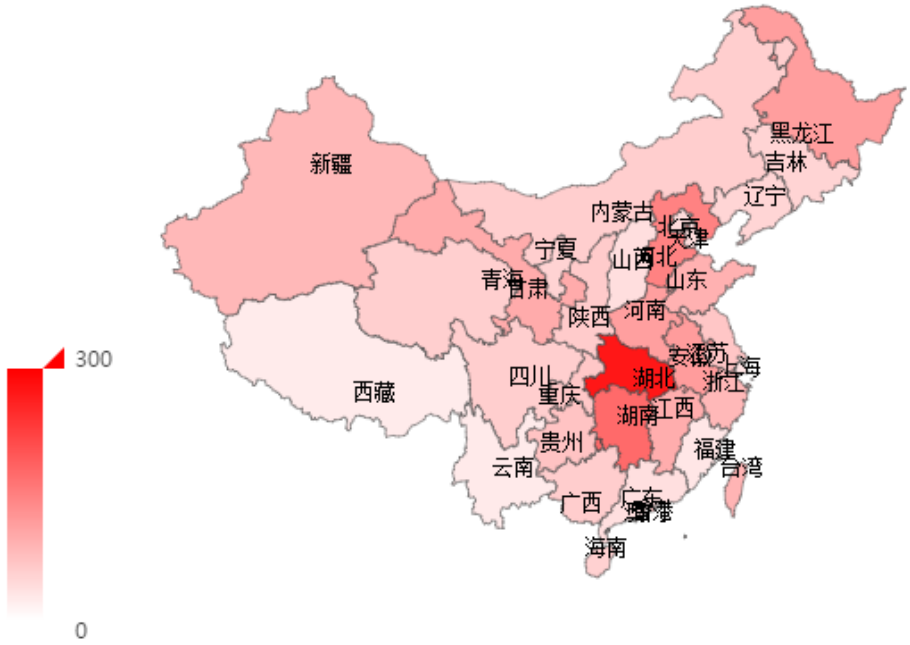
```
var option = {
    series: [
        {
            data: airData,
            geoIndex: 0,
            type: 'map'
        }
    ]
}
```

- 4.结合`visualMap`配合使用

`visualMap`是视觉映射组件, 和之前区域缩放`dataZoom`很类似, 可以做数据的过滤. 只不过`dataZoom`主要使用在直角坐标系的图表, 而`visualMap`主要使用在地图或者饼图中

```
var option = {
    geo: {
        type: 'map',
        map: 'chinaMap',
        roam: true,
        label: {
            show: true
        }
    },
```

```
    series: [
        {
            data: airData,
            geoIndex: 0,
            type: 'map'
        }
    ],
    visualMap: {
        min: 0, // 最小值
        max: 300, // 最大值
        inRange: {
            color: ['white', 'red'] // 颜色的范围
        },
        calculable: true // 是否显示拖拽用的手柄（手柄能拖拽调整选中范围）
    }
}
```



- 地图和散点图结合
  - 1.给`series`这个数组下增加新的对象
  - 2.准备好散点数据,设置给新对象的`data`

```
var scatterData = [
    {
        value: [117.283042, 31.86119] // 散点的坐标，使用的是经纬度
    }
]
```

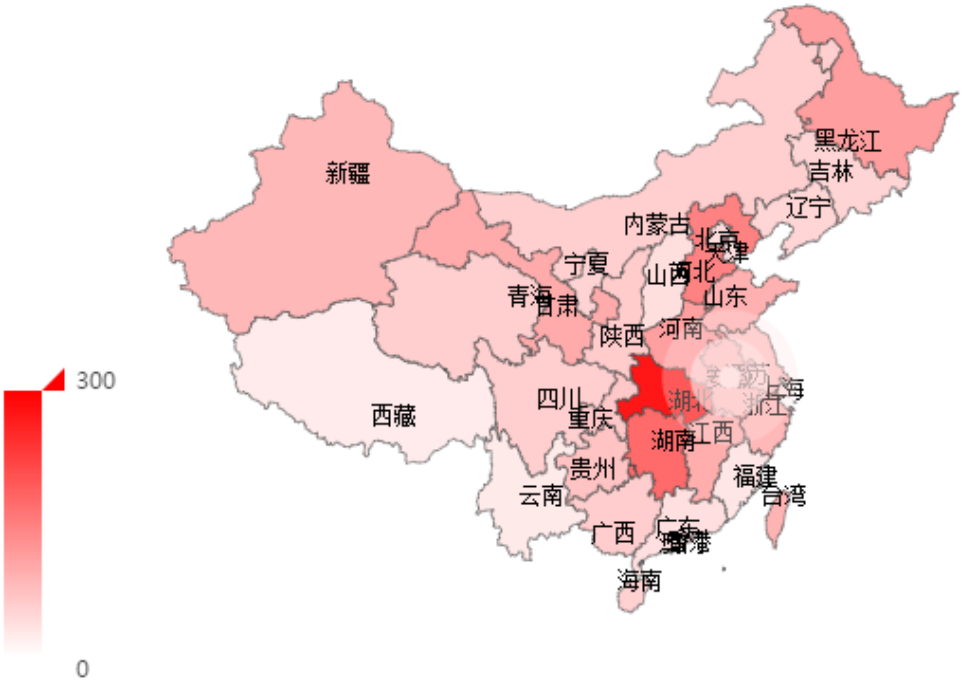
  - 3.配置新对象的`type`

    `type:effectScatter`
  - 让散点图使用地图坐标系统

    `coordinateSystem: 'geo'`
  - 让涟漪的效果更加明显

    `rippleEffect:{ scale: 10 }`

```
var option = {
    series: [
        {
            data: airData,
            geoIndex: 0,
            type: 'map'
        }, {
            data: scatterData,
            type: 'effectScatter',
            coordinateSystem: 'geo',
            rippleEffect: {
                scale: 10
            }
        }
    ]
}
```



3.5.5.地图的特点

地图主要可以帮助我们从宏观的角度快速看出不同地理位置上数据的差异

3.6.图表6 雷达图

3.6.1.雷达图的实现步骤

- 步骤1 `ECharts`最基本的代码结构

```
<!DOCTYPE html>
<html lang="en">
<head>
    <script src="js/echarts.min.js"></script>
</head>
<body>
    <div style="width: 600px;height:400px"></div>
```

```
    <script>
        var mCharts = echarts.init(document.querySelector("div"))
        var option = {}
        mCharts.setOption(option)
    </script>
</body>
</html>
```

此时`option`是一个空空如也的对象

- 步骤2 定义各个维度的最大值

```
var dataMax = [
    {
        name: '易用性', max: 100
    },
    {
        name: '功能', max: 100
    },
    {
        name: '拍照', max: 100
    },
    {
        name: '跑分', max: 100
    },
    {
        name: '续航', max: 100
    }
]
```

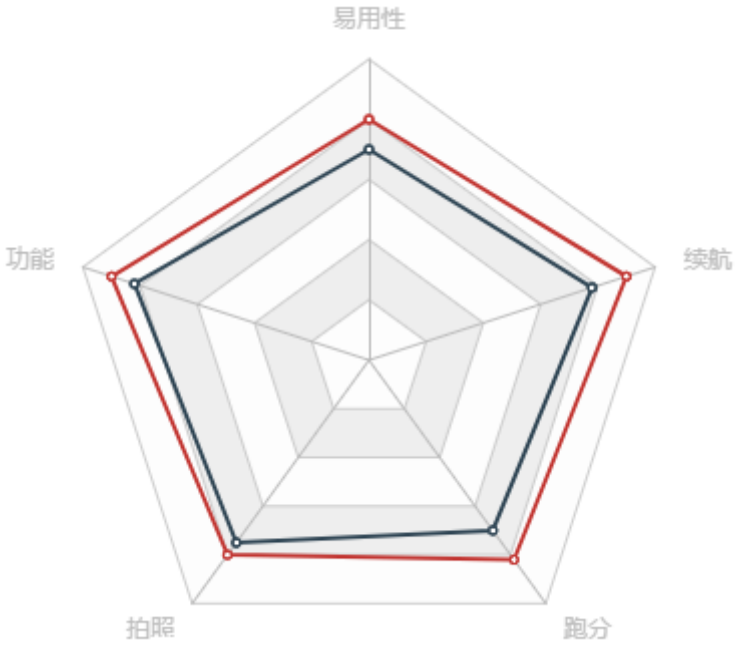
- 步骤3 准备具体产品的数据

```
var hwScore = [80, 90, 80, 82, 90]
var zxScore = [70, 82, 75, 70, 78]
```

- 步骤4 在`series`下设置`type:radar`

```
var option = {
    radar: {
        indicator: dataMax
    },
    series: [
        {
            type: 'radar',
            data: [
                {
                    name: '华为手机1',
                    value: hwScore
                },
                {
                    name: '中兴手机1',
                    value: zxScore
                }
            ]
        }
    ]
```

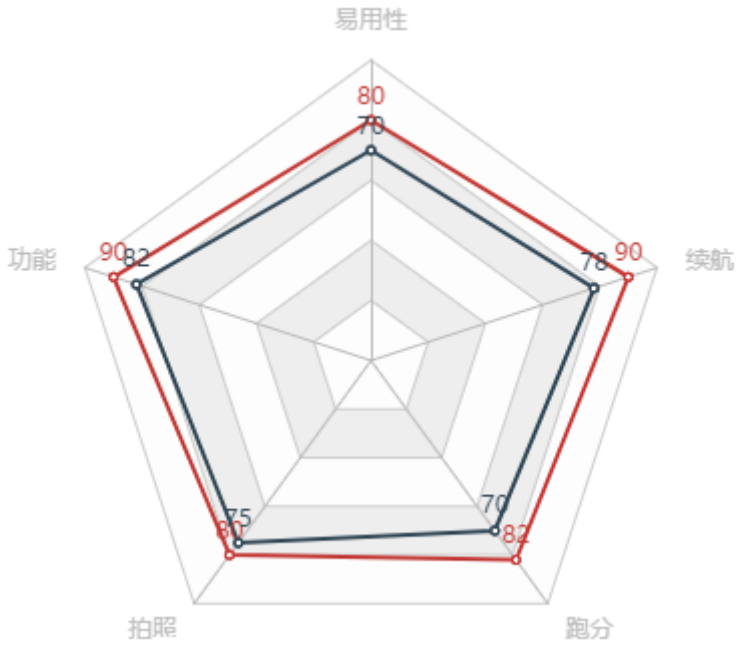
```
}
```



3.6.2.雷达图的常见效果

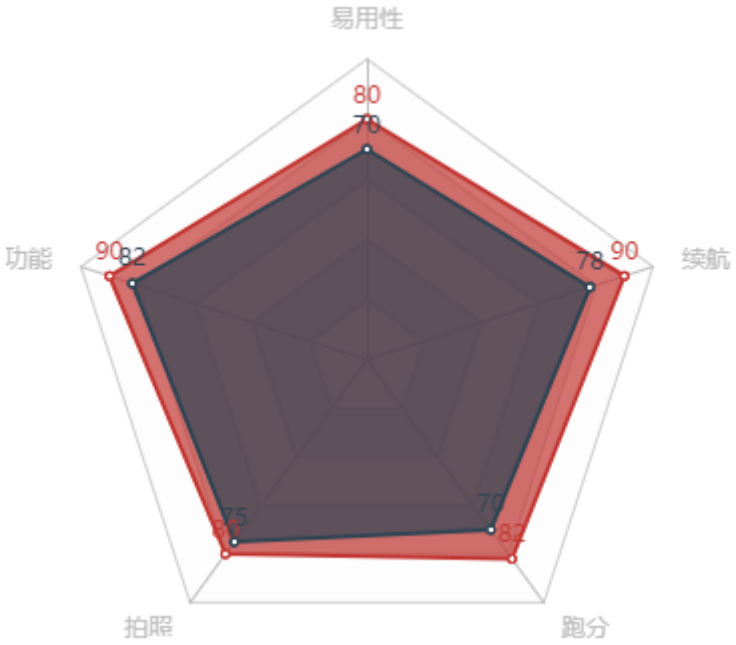
- 显示数值 `label`

```
var option = {
    series: [
        {
            type: 'radar',
            label: {
                show: true
            },
            data: [
                {
                    name: '华为手机1',
                    value: hwScore
                },
                {
                    name: '中兴手机1',
                    value: zxScore
                }
            ]
        }
    ]
}
```



- 区域面积 `areaStyle`

```
var option = {
    series: [
        {
            type: 'radar',
            label: {
                show: true
            },
            areaStyle: {},
            data: [
                {
                    name: '华为手机1',
                    value: hwScore
                },
                {
                    name: '中兴手机1',
                    value: zxScore
                }
            ]
        }
    ]
}
```



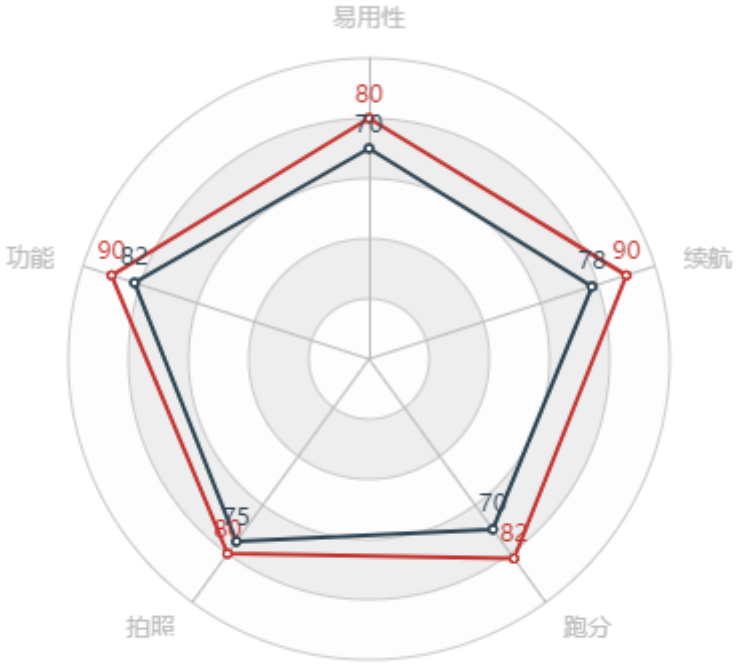
- 绘制类型 `shape`

  雷达图绘制类型，支持 `'polygon'` 和 `'circle'`

  `'polygon'`: 多边形

  `'circle'` 圆形

```
var option = {
  radar: {
    indicator: dataMax,
    shape: 'circle'
  },
  series: [
    {
      type: 'radar',
      label: {
          show: true
      },
      data: [
        {
          name: '华为手机1',
          value: hwScore
        },
        {
          name: '中兴手机1',
          value: zxScore
        }
      ]
    }
  ]
}
```



3.6.3.雷达图的特点

雷达图可以用来分析多个维度的数据与标准数据的对比情况

3.7.图表7 仪表盘图

3.7.1.仪表盘的实现步骤

- 步骤1 `ECharts`最基本的代码结构

```
<!DOCTYPE html>
<html lang="en">
<head>
    <script src="js/echarts.min.js"></script>
</head>
<body>
    <div style="width: 600px;height:400px"></div>
    <script>
        var mCharts = echarts.init(document.querySelector("div"))
        var option = {}
        mCharts.setOption(option)
    </script>
</body>
</html>
```

  此时`option`是一个空空如也的对象

- 步骤2: 准备数据, 设置给`series`下的`data`

  `data:[97]`

- 步骤3: 在`series`下设置 `type:gauge`

```
var option = {
    series: [
        {
            type: 'gauge',
            data: [{
                value: 97,
            }]
        }
    ]
}
```

3.7.2.仪表盘的常见效果

- 数值范围: `max` `min`
- 多个指针: `增加data中数组的元素`
- 多个指针颜色的差异: `itemStyle`

```
var option = {
    series: [
        {
            type: 'gauge',
            data: [{
                value: 97,
                itemStyle: {
                    color: 'pink'
                }
            }, {
                value: 85,
                itemStyle: {
                    color: 'green'
                }
            }],
            min: 50
        }
    ]
}
```

3.7.3.仪表盘的特点

仪表盘可以更直观的表现出某个指标的进度或实际情况

4.配置项小结

4.1.柱状图 bar

series[].type	xAxis	yAxis	markPoint	markLine	label	barWidth
图表类型	x轴	y轴	最大值\最小值	平均值	显示文本	柱宽度

4.2.折线图 line

series[].type	xAxis	yAxis	markPoint	markLine	markArea	smooth
图表类型	x轴	y轴	最大值\最小值	平均值	标注区域	平滑线

lineStyle	areaStyle	boundaryGap	scale
线条风格	风格x轴	紧挨边缘	脱离0值比例

4.3.散点图 scatter

series[].type	xAxis	yAxis	symbolSize
图表类型	x轴	y轴	散点大小

itemStyle	showEffectOn	rippleEffect	scale
散点样式	显示时机	涟漪效果	脱离0值比例

4.4.饼图 pie

series[].type	label	radius	roseType	selectedMode	selectedOffset
图表类型	显示文本	半径	饼图类型	是否多选	选中扇区偏移量

4.5.地图 map

series[].type	geo	map	roam	zoom
图表类型	地理坐标系组件	指明地图数据	开启鼠标拖动和缩放	平均值

center	label	geoIndex	visualMap	coordinateSystem
图表的中心点	是否显示地区	指明关联的geo组件	视觉映射组件	使用坐标系统

4.6.雷达图 radar

series[].type	radar	indicator	label	areaStyle	shape
图表类型	雷达图组件	雷达图的指示器	文字	区域颜色	雷达图形状

4.7.仪表盘 gauge

series[].type	max	min	itemStyle
图表类型	最大值	最小值	指针样式

4.8.直角坐标系配置

- grid

show	borderWidth	borderColor	left	top	right
是否可见	边框宽度	边框颜色	左边	顶部	右边

bottom	width	height
底部	宽度	高度

- axis

type	data	position
轴类型	数据	显示位置

- dataZoom

type	xAxisIndex	yAxisIndex	start	end
缩放块类型	x轴索引	y轴索引	初始值	结束值

4.9.通用配置

- title

textStyle	borderWidth	borderColor	borderRadius
文字样式	边框宽度	边框颜色	边框圆角

left	top	right	bottom
左边	顶部	右边	底部

- tooltip

trigger	triggerOn	formatter
触发类型	触发时机	内容自定义

- toolbox.feature

saveAsImage	dataView	restore	dataZoom	magicType
保存图片	数据视图	重置	缩放	图表转换

- legend

data
图例数据, 需要和series数组中某组数据的name值一致