前言

> *本小册是《千锋大前端小册》系列之 大数据可视化 部分。内容包含E-Chars基础、E-Chars实战及D3基础和进阶。全部内容是依照《千锋教育大前端大纲-大数据可视化》编写。*
>
> *—— 作者：千锋教育·古艺散人*

什么是数据可视化

数据可视化

将结果数据的每一个数据项，作为单个图元元素展示，大量的数据集构成数据图像，同时将数据的各个属性以多维度的方式展现，从而提高数据的可读性

数据可视化的优点

图形化展示比文字的描述能力更强，降低大数据据阅读门槛，清晰有效地传达与沟通信息。具体做法：是指将大型数据集中的数据以图形图像形式表示。基于几何的技术、面向像素技术、基于图标的技术、基于层次的技术、基于图像的技术和分布式技术等等

1. 国内经典案例：
2. 百度统计（https://tongji.baidu.com/）（导航栏demo链接进入）
3. 阿里云 数加（https://data.aliyun.com/）
4. 北京数字冰雹（http://www.uipower.com/）

数据处理流程

1. 将已经存在的数据管理起来（ 收集、采集 ）
2. 数据清洗（etl extract transform load 数据仓库技术 ）
3. 数据计算：统计分析(分组、极值、多维度展示)， 结果预测（spark）、python机器学习（分类/聚类算法）
4. 数据可视化
    - echarts：(百度开源项目，国内应用领域十分广泛，apache孵化器，各大领域，入门比较容易、主要是修改配置)
    - highCharts： （学习可以，商用需要授权，09年发布，使用纯js编写的图表库）

- d3：D3 的全称是（Data-Driven Documents），一个被数据驱动的文档。听名字有点抽象，说简单一点，其实就是一个 JavaScript 的函数库，主要是用来做数据可视化，将强大的可视化组件应用于需求中。

预备知识

1. HTML：超文本标记语言，用于设定网页的内容
2. CSS：层叠样式表，用于设定网页的样式
3. JavaScript：一种直译式脚本语言，用于设定网页的行为
4. DOM：文档对象模型，用于修改文档的内容和结构
5. SVG：可缩放矢量图形，用于绘制可视化的图形

ECharts 轻松上手

1. Echarts：一个纯 Javascript 的图表库，而且ECharts 3 开始不再强制使用 AMD 的方式按需引入，
2. 代码里也不再内置 AMD 加载器。只需要像普通的 JavaScript 库一样用 script 标签引入。
3.
4. js下载：
5. http://echarts.baidu.com/download.html 。(可定制)
6.
7. ui皮肤定制：
8. https://echarts.baidu.com/theme-builder/

ECharts 起步

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

1. 在绘图前我们需要为 ECharts 准备一个具备高宽的 DOM 容器

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

1.

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>start</title>
  <script src="./echarts.min.js"></script>
  <style>
```

```
    #container {
    width: 600px;
    height: 500px;
    }
    </style>
</head>

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

    <script>
    var myChart = echarts.init(document.getElementById('container'));
    var option = {
    //标题组件，主标题，副标题
    title: {
    text: 'echarts 入门',
    link: 'https://www.baidu.com',
    subtext: 'demo'
    },
    //图例组件
    legend: {
    data: ['销量'],
    show: true,
    right: '20'
    },
    //x坐标轴
    xAxis: {
    data: ["衬衫", "羊毛衫", "雪纺衫", "裤子", "高跟鞋", "袜子"]
    },
    //y坐标轴
    yAxis: {
    show: true
    },
    //通过type指定表的类型
    series: [
    {
    name: '销量',
    type: 'bar',
    data: [10, 30, 25, 5, 40, 35],
    animationDelay: function (idx) {
```

```
    animationDelay: function (idx) {
    return idx * 10;
    }
    }
    ]
    }
    myChart.setOption(option);
    </script>

</body>

</html>
```

常用配置介绍

1. 参见：https://www.echartsjs.com/zh/option.html

Vue + Echart 项目实战

安装脚手架

```
yarn global add @vue/cli
```

创建项目

```
vue init vue-echarts
```

配置vue.config.js

在根目录下创建 vue.config.js：

```
module.exports = {
 publicPath: '/',
 outputDir: 'dist',
 configureWebpack: {
 devtool: 'source-map',
 externals: {

```

```
  }
  },
  devServer: {
  proxy: {
  '/api': {
  target: 'http://localhost:9000',
  changeOrigin: true,
  pathRewrite: {
  '^/api': ''
  }
  }
  }
  }
 }
```

安装echarts

```
yarn add echarts
```

一些模拟数据

柱状图模拟数据 (bar.json)

```
{
 "data": {
 "name": [
 "衬衫",
 "羊毛衫",
 "雪纺衫",
 "裤子",
 "高跟鞋",
 "袜子"
 ],
 "sales": [
 10,
 30,
 25,
 5,
 40,
 35
```

```
  ],
  "cost": [
  5,
  3,
  6,
  0.5,
  4,
  15
  ]
  }
}
```

饼图模拟数据 （pie.json）

```
 {
  "data": {
  "title": "某站点用户访问来源",
  "list": [
   {
  "value": 335,
  "name": "直接访问"
  },
   {
  "value": 310,
  "name": "邮件营销"
  },
   {
  "value": 234,
  "name": "联盟广告"
  },
   {
  "value": 135,
  "name": "视频广告"
  },
   {
  "value": 1548,
  "name": "搜索引擎"
  }
  ]
  }
 }
```

json-server 模拟接口

安装 json-server

```
yarn global add json-server
```

创建json-server运行环境

在项目根目录下创建mock文件夹，将上述的.json模拟数据放进去。再创建一个mock.js文件：

```
// /mock/mock.js
module.exports = () => {
 return {
 pie: require('./pie.json'),
 bar: require('./bar.json')
 }
}
```

在 package.json 中的 script 中添加脚本：

```
"mock": "json-server ./mock/mock.js --watch"
```

App.vue

```
<template>
 <div id="app">
 <Pie/>
 </div>
</template>

<script>
import Pie from "./Pie"

export default {
 name: "app",
 components: {
 Pie
 },
 data() {
 return {}
 }
}
```

```
</script>

<style lang="scss">
#app {
 text-align: center;
 color: #2c3e50;
 margin-top: 60px;
 width: 600px;
 height: 500px;
}
</style>
```

Pie.vue

```
<template>
 <div ref="container" style="width:500px;height:500px;">
 111
 </div>
</template>

<script>
import echarts from "echarts"

export default {
 name: "app",
 created() {},
 data() {
 return {
 title: "某站点用户访问来源",
 data: []
 }
 },
 components: {},
 methods: {
 initBar() {
 var myChart = echarts.init(this.$refs.container)
 var option = {
 //标题组件，主标题，副标题
 title: {
 text: "echarts 入门",
```

```
27.   link: "https://www.baidu.com",
28.   subtext: "demo"
29.   },
30.   //图例组件
31.   legend: {
32.   data: ["销量"],
33.   show: true,
34.   right: "20"
35.   },
36.   //x坐标轴
37.   xAxis: {
38.   data: this.name,
39.   show: false
40.   },
41.   //y坐标轴
42.   yAxis: {
43.   show: true,
44.   show: false
45.   },
46.   //系列，通过type指定表的类型
47.   series: [
48.   {
49.   name: "销量",
50.   type: "pie",
51.   data: this.data,
52.   radius: "55%",
53.   center: ["40%", "50%"]
54.   }
55.   ]
56.   }
57.   myChart.setOption(option)
58.   }
59.   },
60.   mounted() {
61.   fetch("http://localhost:3000/data")
62.   .then(res => res.json())
63.   .then(res => {
64.   this.title = res.title
65.   this.data = res.list
66.   this.initBar()
67.   })
```

```
  }
 }
</script>

<style lang="scss">
#app {
 text-align: center;
 color: #2c3e50;
 margin-top: 60px;
 width: 600px;
 height: 500px;
}
</style>
```

echarts 其它问题

折线图的圆滑展示，和曲线的差异配置，文本样式设置

```
smooth
```

如何避免折线数据堆叠的问题？

```
stack
```

动画：柱状图动画延迟，animationDelay

```
 series: [
 {
 name: "销量",
 type: "bar",
 data: this.sales,
 animationDelay: function(idx) {
 return idx * 10
 }
 },
 {
 name: "成本",
 type: "bar",
 data: this.cost,
 animationDelay: function(idx) {
 return idx * 500
 }
```

```
  }
 ]
animationEasing: "elasticOut",
animationDelayUpdate: function(idx) {
 return idx * 50
}
```

外部数据更新

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

使用vue-echarts

前往 vue-echarts 官网

安装 vue-echarts

```
yarn add vue-echarts --dev
```

配置 vue.config.js

Vue-ECharts 默认在 webpack 环境下会引入未编译的源码版本，如果你正在使用官方的 Vue CLI 来创建项目，可能会遇到默认配置把 node_modules 中的文件排除在 Babel 转译范围以外的问题。请按如下方法修改配置：

当使用 Vue CLI 3+ 时，需要在 vue.config.js 中的 transpileDependencies 增加 vue-echarts 及 resize-detector，如下：

```
// vue.config.js
module.exports = {
 transpileDependencies: [
 'vue-echarts',
 'resize-detector'
 ]
}
```

创建 Polar.vue 组件

在 /src/component/Polar.vue

```
<template>
 <v-chart :options="polar"/>
</template>

```

```
<style>
/**
 * 默认尺寸为 600px×400px，如果想让图表响应尺寸变化，可以像下面这样
 * 把尺寸设为百分比值（同时请记得为容器设置尺寸）。
 */
.echarts {
 width: 100%;
 height: 100%;
}
</style>

<script>
import ECharts from 'vue-echarts'
import 'echarts/lib/chart/line'
import 'echarts/lib/component/polar'

export default {
 components: {
 'v-chart': ECharts
 },
 data () {
 let data = []

 for (let i = 0; i <= 360; i++) {
 let t = i / 180 * Math.PI
 let r = Math.sin(2 * t) * Math.cos(2 * t)
 data.push([r, i])
 }

 return {
 polar: {
 title: {
 text: '极坐标双数值轴'
 },
 legend: {
 data: ['line']
 },
 polar: {
 center: ['50%', '54%']
 },
 tooltip: {
```

```
  trigger: 'axis',
  axisPointer: {
  type: 'cross'
  }
  },
  angleAxis: {
  type: 'value',
  startAngle: 0
  },
  radiusAxis: {
  min: 0
  },
  series: [
  {
  coordinateSystem: 'polar',
  name: 'line',
  type: 'line',
  showSymbol: false,
  data: data
  }
  ],
  animationDuration: 2000
  }
  }
  }
</script>
```

D3快速入门

D3安装

```
<script src="https://d3js.org/d3.v5.min.js"></script>
```

选择器

```
在 D3 中，用于选择元素的函数有两个，这两个函数返回的结果称为选择集。
d3.select()：选择所有指定元素的第一个
d3.selectAll()：选择指定全部元素
例如，选择集的常见用法如下。
var body = d3.select("body"); //选择文档中的body元素
```

```
var body = d3.select("body"); //选择文档中的body元素
var p1 = body.select("p"); //选择body中的第一个p元素
var p = body.selectAll("p"); //选择body中的所有p元素
var svg = body.select("svg"); //选择body中的svg元素
var rects = svg.selectAll("rect"); //选择svg中所有的rect元素
var id = body.select("#id"); //选择body中id元素
var class = body.select(".class");//选择body中class类元素
```

链式操作：

```
d3.select("#container").text("1000phone").attr("font-size","12px");
```

绑定数据

选择集和绑定数据通常是一起使用的，D3 中是通过以下两个函数来绑定数据的：

1. datum()：绑定一个数据到选择集上
2. data()：绑定一个数组到选择集上，数组的各项值分别与选择集的各元素绑定

假设现在有三个段落元素如下：

```
<p></p>
<p></p>
<p></p>
```

对于datum()：

假设有一字符串 逆战2020，要将此字符串分别与三个段落元素绑定，代码如下：

```
var data = '逆战2020';
var container = d3.select("#app");
container.selectAll('p')
 .datum(data)
 .text(function (d, i) {
 return "第 " + i + " 个元素绑定的数据是： " + d;
 })
```

绑定数据后，使用此数据来修改三个段落元素的内容，其结果如下：

```
第 0 个元素绑定的数据是： 逆战2020

第 1 个元素绑定的数据是： 逆战2020

第 2 个元素绑定的数据是： 逆战2020
```

对于data()：

有一个数组，接下来要分别将数组的各元素绑定到三个段落元素上。

```
var datalist = [10, 20, 30];
```

调用 data() 绑定数据，并替换三个段落元素的字符串为被绑定的字符串，代码如下

调用 data() 绑定数据，并替换二个段落元素的子符串为被绑定的子符串，代码如下：

```
var datalist = [10, 20, 30];
var container = d3.select("#app");
//更新数据
container.selectAll('p')
 //绑定数据源
 .data(datalist)
 .text(function (data, index) {
 return data;
 })
```

结果自然是三个段落的文字分别变成了数组的三个字符串。

```
10
20
30
```

前面代码也用到了一个无名函数 function(d, i)，其对应的情况如下：

```
d ------- data 数据
i ------- index 索引
```

当 i == 0 时， d 为 10。
当 i == 1 时， d 为 20。
当 i == 2 时， d 为 30。
此时，三个段落元素与数组 dataset 的三个字符串是一一对应的，在函数 function(d, i) 直接 return d 即可。

选择、插入、删除元素

1.选择元素

```
<p>10</p>
<p>20</p>
<p>30</p>
```

选择第一个元素

```
d3.select("body").select("p").style("color","red");
```

选择第所有元素

```
d3.select("body").selectAll("p").style("color","red");
```

选择第二个元素

```
<p id="second">20</p>
d3.select("#second").style("color","red");
```

选择后两个元素,给后两个元素添加 class，

```
<p class="myclass">Moon</p>
<p class="myclass">You</p>
```

由于需要选择多个元素，要用 selectAll。

```
d3.selectAll(".myclass").style("color","red")
```

插入元素
插入元素涉及的函数有两个：

append()：在选择集末尾插入元素
insert()：在选择集前面插入元素
假设有三个段落元素，与上文相同。

append()

```
d3.select("body").append("p").text("Star");
```

insert

```
d3.select("body").insert("p","#second").text("20");
```

删除元素

```
d3.select("#second").remove();
```

理解 update()、enter()、exit()

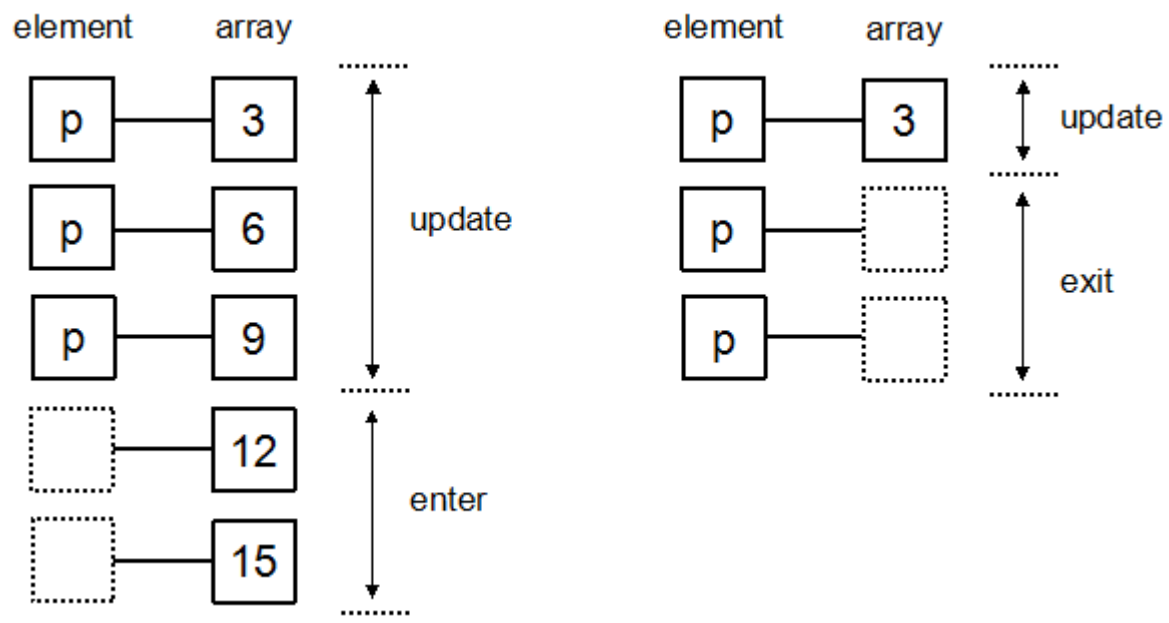
数据绑定的时候可能出现 DOM 元素与数据元素个数不匹配的问题，那么 enter 和 exit 就是用来处理这个问题的。enter 操作用来添加新的 DOM 元素，exit 操作用来移除多余的 DOM 元素

Update、Enter、Exit 是 D3 中三个非常重要的概念，它处理的是当选择集和数据的数量关系不确定的情况。

1. update(), 当对应的元素正好满足时 （ 绑定数据数量 = 对应元素 ），实际上并不存在这样一个函数，只是为了要与之后的 enter 和 exit 一起说明才想象有这样一个函数。但对应元素正好满足时，直接操作即可，后面直接跟 text ，style 等操作即可。
2. enter(), 当对应的元素不足时 （ 绑定数据数量 > 对应元素 ），当对应的元素不足时，通常要添加元素，使之与绑定数据的数量相等。后面通常先跟 append 操作。
3. exit(), 当对应的元素过多时 （ 绑定数据数量 < 对应元素 ），当对应的元素过多时，通常要删除元素，使之与绑定数据的数量相等。后面通常要跟 remove 操作。

如果数组为 [3, 6, 9, 12, 15]，将此数组绑定到三个 p 元素的选择集上。可以想象，会有两个数据没有元素与之对应，这时候 D3 会建立两个空的元素与数据对应，这一部分就称为 Enter。而有元素与数据对应的部分称为 Update。如果数组为 [3]，则会有两个元素没有数据绑定，那么没有数据绑定的部分被称为 Exit。示意图如下所示

exit。示意图如下所示。



而如下代码的意思是：此时 SVG 里没有 rect 元素，即元素数量为 0。有一数组 dataset，将数组与元素数量为 0 的选择集绑定后，选择其 Enter 部分（请仔细看上图），然后添加（append）元素，也就是添加足够的元素，使得每一个数据都有元素与之对应。

```
svg.selectAll("rect") //选择svg内所有的矩形
 .data(dataset) //绑定数组
 .enter() //指定选择集的enter部分
 .append("rect") //添加足够数量的矩形元素
```

1. Update和Enter的使用

当对应的元素不足时 （ 绑定数据数量 > 对应元素 ），需要添加元素（append）。
现在 body 中有三个 p 元素，要绑定一个长度大于 3 的数组到 p 的选择集上，然后分别处理 update 和 enter 两部分。

```
var dataset = [ 3 , 6 , 9 , 12 , 15 ];

//选择body中的p元素
var p = d3.select("body").selectAll("p");

//获取update部分
var update = p.data(dataset);

//获取enter部分
var enter = update.enter();
```

```
11.
12.  //update部分的处理：更新属性值
13.  update.text(function(d){
14.   return "update " + d;
15.  });
16.
17.  //enter部分的处理：添加元素后赋予属性值
18.  enter.append("p")
19.   .text(function(d){
20.   return "enter " + d;
21.   });
```

页面效果：

```
1.  enter 3
2.  enter 6
3.  enter 9
4.  enter 12
5.  enter 15
```

需要注意的是：

update 部分的处理办法一般是：更新属性值
enter 部分的处理办法一般是：添加元素后，赋予属性值

2. Update和Exit的使用

当对应的元素过多时 （ 绑定数据数量 < 对应元素 ），需要删掉多余的元素。
现在 body 中有三个 p 元素，要绑定一个长度小于 3 的数组到 p 的选择集上，然后分别处理 update 和 exit 两部分。

```
1.  var dataset = [ 3 ];
2.
3.  //选择body中的p元素
4.  var p = d3.select("body").selectAll("p");
5.
6.  //获取update部分
7.  var update = p.data(dataset);
8.
9.  //获取exit部分
10. var exit = update.exit();
11.
12. //update部分的处理：更新属性值
13. update.text(function(d){
```

```
 return "update " + d;
});

//exit部分的处理：修改p元素的属性
exit.text(function(d){
 return "exit";
});

//exit部分的处理通常是删除元素
// exit.remove();
```

需要注意的是:
exit 部分的处理办法一般是：删除元素(remove)

过渡

D3 支持动画效果，这种动画效果可以通过对样式属性的过渡实现。其补间插值支持多种方式，比如线性、弹性等。此外 D3 内置了多种插值方式，比如对数值类型、字符类型路径数据以及颜色等。

启动过渡效果，与以下四个方法相关:

//创建一个过渡对象。但是由于每个选择集中都有transition()方法，可用
d3.select(“rect”).transition()的方式来创建过渡，因此一般不直接用d3.transition()。
d3.transition([selection],[name])

//设定延迟的时间。过渡会经过一定时间后才开始发生。单位是毫秒。
transition.delay([delay])

//设定过渡的持续时间(不包括延迟时间)，单位是毫秒。如:duration(2000),是持续2000ms。
transition.duration([duration])

//设定过渡样式，例如线性过渡、在目标处弹跳几次等方式。
transition.ease(vlaue[,arguments])

接下来制作一个过渡效果:

```
var width = 600;
var height = 400;

var svg = d3.select("#body")
 .append("svg")
 .attr("width",width)
 .attr("height",height)

```

```
svg.append("rect")
  .attr("fill","yellow")
  .attr("x",100)
  .attr("y",100)
  .attr("width",100)
  .attr("height",30)
  .transition()
  .duration(750)
  .delay(function(d, i) { return i * 10; })
  .attr("width",300)
  .attr("height",300)
```

除了 D3 提供的过渡之外，你也可以通过 CSS 动画来实现对元素的过渡效果。

做一个简单的图表

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>Document</title>
  <script src="./d3.v5.min.js"></script>
  <style>
  #app {
  margin: 0 auto;
  width: 500px;
  height: 400px;
  background: #efefef;
  position: relative;
  }

  .bar {
  width: 30px;
  /* height: 50px; */
  background: green;
  position: absolute;
  bottom: 100px;
  }

```

```
  .bar span {
  display: block;
  text-align: center;
  }
  </style>
</head>

<body>
  <div id="app">
  <!-- <div class="bar"></div> -->
  </div>

  <script>

  var datalist = [10, 20, 30, 40, 50];
  var container = d3.select("#app");

  container.selectAll('div')
  .data(datalist)
  .enter()
  .append('div')
  .classed('bar', true)
  .style('height', function (d, i) {
  return d * 5 + 'px';
  })
  .style('left', function (d, i) {
  return i * 35 + 'px';
  })
  .append('span')
  .text(function (d) {
  return d;
  })
  .style('color', function (d) {
  if (d > 30) {
  return 'red';
  }
  })
  </script>

</body>
```

```
</html>
```

svg基础

https://developer.mozilla.org/zh-CN/docs/Web/SVG
https://www.d3js.org.cn/svg/get_start/

常用标签

```
Rect Ellipse Line Circle polygon polyline
path Text defs g use Animate
<svg width="500" height="500" style="background: #efefef;">

  <!-- <rect x="50" y="100" width="100" height="50" fill="red" style="" stroke="blue" stroke-width="5" /> -->

  <!-- <ellipse cx="200" cy="200" rx="50" ry="100" style="fill:orange; stroke: orangered; stroke-width: 5px;" /> -->

  <!-- <line x1="50" y1="50" x2="450" y2="450" stroke="red" stroke-width="3" /> -->


  <!-- <circle cx="225" cy="225" r="100" style="fill:peru;" /> -->

  <!-- <polygon points="50,20 150,60 120,200 100,200" style="fill:pink;" fill="none" stroke="red" stroke-width="5" /> -->

  <!-- <!-- <polyline points="10,20 50,60 120,200 200,300" fill="blue" stroke="red" stroke-width="5"></polyline> -->
  -->

  <!-- <path d="M30,30 L200,200 L230,260" fill="none" stroke="green" stroke-width="5" />

  -->

  <!-- <defs>

  <g id="group">
  <rect x="50" y="100" width="100" height="50" style="fill:green;" stroke="blue" stroke-width="5">

```

```
  <animate attributeName="opacity" from="1" to="0" dur="5s" repeatCount="indefinite" />
  </rect>
  <circle cx="225" cy="225" r="100" style="fill:peru;">

  <animate attributeName="cx" from="225" to="100" dur="5s" repeatCount="indefinite" />
  </circle>
  </g>

  </defs> -->

  <!-- <use xlink:href="#group" x="30" y="30" />
  <use xlink:href="#group" x="130" y="130" /> -->

  <!-- <text x="200" y="200" style="fill: none; stroke: red; stroke-width: 1; font-size: 45px;">NZGP1916</text> -->


  <clipPath id="myClipPath">
  <rect width="200" height="100" x="200" y="200"></rect>
  </clipPath>

  <circle cx="260" clip-path="url(#myClipPath)" cy="260" r="100" style="" />

  </svg>
```

将图表标签更换成svg

```
 <!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>Document</title>
  <script src="./d3.v5.min.js"></script>
  <style>
  #app {
  margin: 0 auto;
  width: 500px;
```

```
14.   height: 400px;
15.   background: #efefef;
16.   position: relative;
17.   }
18.
19.   .bar {
20.   width: 30px;
21.   /* height: 50px; */
22.   fill: green;
23.   position: absolute;
24.   bottom: 100px;
25.   }
26.
27.   .bar span {
28.   display: block;
29.   text-align: center;
30.   }
31.   </style>
32. </head>
33.
34. <body>
35.   <svg id="app" style="width: 500px; height:400;">
36.   <!-- <div class="bar"></div> -->
37.   </svg>
38.
39.   <script>
40.
41.   var datalist = [10, 20, 30, 40, 50];
42.   var container = d3.select("#app");
43.
44.   container.selectAll('rect')
45.   .data(datalist)
46.   .enter()
47.   .append('rect')
48.   .classed('bar', true)
49.   .style('height', function (d, i) {
50.   return d * 5 + 'px';
51.   })
52.   .attr('x', function (d, i) {
53.   return i * 35 + 'px';
54.   })
```

```
55.   .attr('y', function (d, i) {
56.   return 400 - d * 5 - 20 + 'px';
57.   })
58.   .style('color', function (d) {
59.   if (d > 30) {
60.   return 'red';
61.   }
62.   })
63.
64.
65.
66.   container.selectAll('text')
67.   .data(datalist)
68.   .enter()
69.   .append('text')
70.   .attr('text-anchor', 'middle')
71.   .text(function (d) {
72.   return d;
73.   })
74.   .attr('x', function (d, i) {
75.   return i * 35 + 15 + 'px';
76.   })
77.   .attr('y', function (d, i) {
78.   return 400 - d * 5 - 20 + 'px';
79.   })
80.   .style('color', function (d) {
81.   if (d > 30) {
82.   return 'red';
83.   }
84.   })
85.
86.   </script>
87.
88. </body>
89.
90. </html>
```

D3进阶

加载外部数据

加载csv数据，可以通过d3.csv(data,function) 进行操作
加载json数据，跟上面是一样的，只需要将csv改为json就行

```
d3.csv('./data.csv').then((result) => {
var container = d3.select("#app");
container.selectAll('h1')
.data(result.columns)
.enter()
.append('h1')
.text(function (d) {
return d;
})
})
```

data.csv

```
10,20,30,40,50,60
```

更多请参考

比例尺的使用

D3中有个重要的概念就是比例尺。比例尺就是把一组输入域映射到输出域的函数。映射就是两个数据集之间元素相互对应的关系。比如输入是1，输出是100，输入是5，输出是10000，那么这其中的映射关系就是你所定义的比例尺。

D3中有各种比例尺函数，有连续性的，有非连续性的，本文对于常用比例尺进行一一介绍。

1. d3.scaleLinear() 线性比例尺

使用d3.scaleLinear()创造一个线性比例尺，而domain()是输入域，range()是输出域，相当于将domain中的数据集映射到range的数据集中。

```
let scale = d3.scaleLinear().domain([1,5]).range([0,100])
```

映射关系：

接下来，我们来研究这个比例尺的输入和输出。

```
scale(1) // 输出:0
scale(4) // 输出:75
scale(5) // 输出:100
```

刚才的输入都是使用了domain区域里的数据，那么使用区域外的数据会得出什么结果呢？

```
scale(-1) // 输出:-50
scale(10) // 输出:225
```

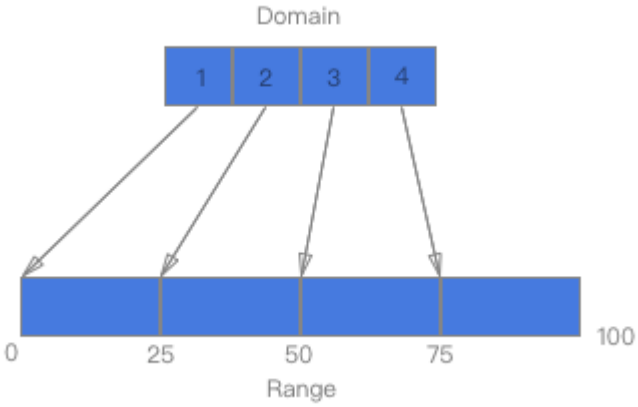
所以这只是定义了一个映射规则，映射的输入值并不局限于domain()中的输入域。

2. d3.scaleBand() 序数比例尺

d3.scaleBand()并不是一个连续性的比例尺，domain()中使用一个数组，不过range()需要是一个连续域。

```
let scale = d3.scaleBand().domain([1,2,3,4]).range([0,100])
```

映射关系：



看一下输入与输出：

```
scale(1) // 输出:0
scale(2) // 输出:25
scale(4) // 输出:75
```

当输入不是domain()中的数据集时：

```
scale(0) // 输出:undefined
scale(10) // 输出:undefined
```

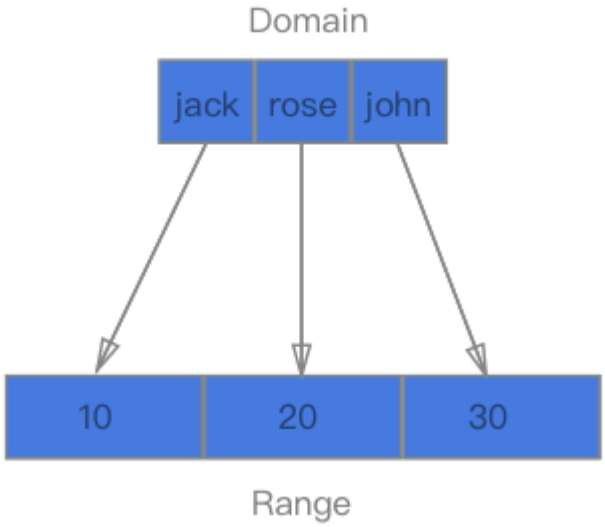
由此可见，d3.scaleBand()只针对domain()中的数据集映射相应的值。

3. d3.scaleOrdinal() 序数比例尺

d3.scaleOrdinal()的输入域和输出域都使用离散的数据。

```
let scale = d3.scaleOrdinal().domain(['jack', 'rose', 'john']).range([10, 20, 30])
```

映射关系：



输入与输出：

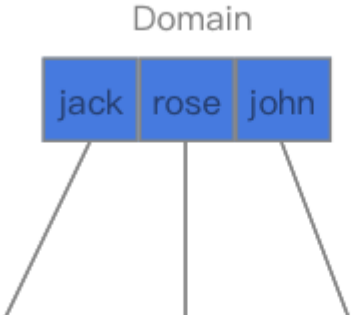
```
scale('jack') // 输出:10
scale('rose') // 输出:20
scale('john') // 输出:30
```

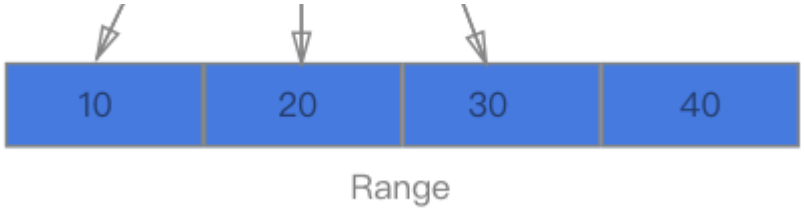
当输入不是domain()中的数据集时：

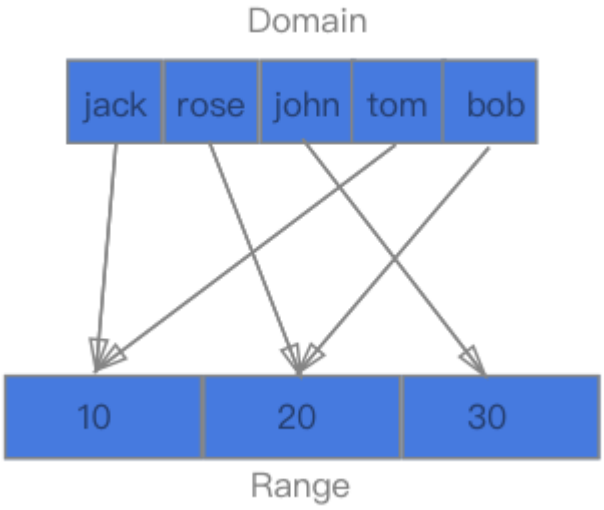
```
scale('tom') // 输出:10
scale('trump') // 输出:20
```

输入不相关的数据依然可以输出值。所以在使用时，要注意输入数据的正确性。

我们从上面的映射关系中可以看出，domain()和range()的数据是一一对应的，如果两边的值不一样呢？下面两张图说明这个问题：







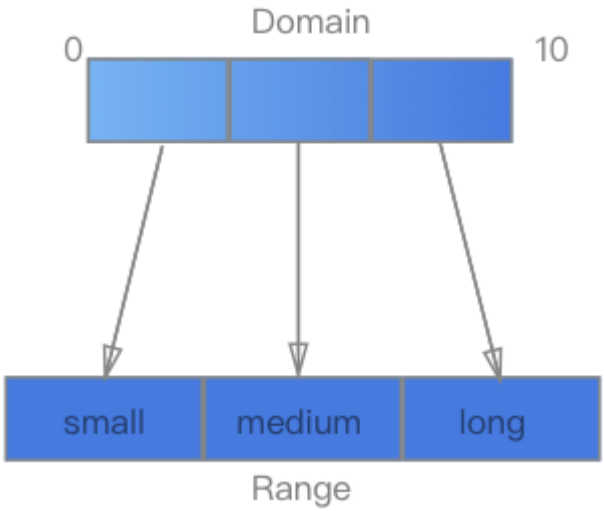
domain()的值按照顺序循环依次对应range()的值。

4. d3.scaleQuantize() 量化比例尺

d3.scaleQuantize()也属于连续性比例尺。定义域是连续的，而输出域是离散的。

```
1.  let scale = d3.scaleQuantize().domain([0, 10]).range(['small', 'medium', 'long'])
```

映射关系：



输入与输出：

```
scale(1) // 输出:small
scale(5.5) // 输出:medium
scale(8) // 输出:long
```

而对于domain()域外的情况：

```
scale(-10) // 输出：small
scale(10) // 输出：long
```

大概就是对于domain()域的两侧的延展。

5. d3.scaleTime() 时间比例尺

d3.scaleTime()类似于d3.scaleLinear()线性比例尺，只不过输入域变成了一个时间轴。

```
let scale = d3.scaleTime()
 .domain([new Date(2020, 2, 20, 0), new Date(2020, 2, 20, 2)])
 .range([0,100])
```

输入与输出：

```
scale(new Date(2020, 2, 20, 0)) // 输出:0
scale(new Date(2020, 2, 20, 1)) // 输出:50
```

时间比例尺较多用在根据时间顺序变化的数据上。另外有一个d3.scaleUtc()是依据世界标准时间(UTC)来计算的。

6. 颜色比例尺

D3提供了一些颜色比例尺，10就是10种颜色，20就是20种：

```
d3.schemeCategory10
d3.schemeCategory20
d3.schemeCategory20b
d3.schemeCategory20c
```

```
// 定义一个序数颜色比例尺
let color = d3.scaleOrdinal(d3.schemeCategory10)
```

7. 其他比例尺

另外有一些函数比例尺的功能，从名称上就可见一斑。

```
d3.scaleIdentity() // 恒等比例尺
d3.scaleSqrt() // 乘方比例尺
d3.scalePow() // 类似scaleSqrt的乘方比例尺
d3.scaleLog() // 对数比例尺
```

```
d3.scaleQuantile() // 分位数比例尺
```

8. invert()与invertExtent()方法

上述的各种使用比例尺的例子都相当于一个正序的过程，从domain的数据集映射到range数据集中，那么有没有逆序的过程呢？D3中提供了invert()以及invertExtent()方法可以实现这个过程。

```
let scale = d3.scaleLinear().domain([1,5]).range([0,100])
scale.invert(50) // 输出:3
let scale2 = d3.scaleQuantize().domain([0,10]).range(['small', 'big'])
scale2.invertExtent('small') // 输出:[0,5]
```

不过，值得注意的是，这两种方法只针对连续性比例尺有效，即domain()域为连续性数据集的比例尺。那么非连续性的比例尺就没有invert()方法了吗？

收尾

到此，对于D3V4版本中的常见比例尺的映射关系都进行了介绍，而各个比例尺还提供了许多其他功能，比如在绘制坐标轴中用到的ticks()，tickFormat()等功能，具体API可以参见此处。关于第8点最后提出的问题，请听下回分解。

给柱形图添加比例尺

```
<script>
 var datalist = [10, 20, 30, 40, 50];

 const linear = d3.scaleLinear()
 .domain([0, d3.max(datalist)])
 .range([0, 400]);

 var container = d3.select("#app");

 container.selectAll('rect')
 .data(datalist)
 .enter()
 .append('rect')
 .classed('bar', true)
 .style('height', function (d, i) {
 return linear(d) - 10 + 'px';
 })
 .attr('x', function (d, i) {
 return i * 35 + 'px';
 })
 .attr('width', function (d, i) {
```

```
return 30 + 'px';
})
.attr('y', function (d, i) {
return 400 - linear(d) + 5 + 'px';
})
.append('text')
.text(function (d) {
return d;
})
.style('color', function (d) {
if (d > 30) {
return 'red';
}
})
</script>
```

进一步完善，添加序数比例尺：

```
<script>

var datalist = [5, 10, 20, 30, 40, 50];
//线性比例尺
const linear = d3.scaleLinear()
.domain([2, d3.max(datalist)])
.range([2, 398]);

//序数比例尺
const scaleBand = d3.scaleBand()
.domain(d3.range(0, datalist.length))
.range([2, 498])
.paddingInner(0.05);

var container = d3.select("#app");

container.selectAll('rect')
.data(datalist)
.enter()
.append('rect')
.classed('bar', true)
.style('height', function (d, i) {
return linear(d) - 10 + 'px';
})
```

```
25.   .attr('x', function (d, i) {
26.   return scaleBand(i) + 'px';
27.   })
28.   .attr('width', function (d, i) {
29.   return scaleBand.bandwidth() + 'px';
30.   })
31.   .attr('y', function (d, i) {
32.   return 400 - linear(d) + 5 + 'px';
33.   })
34.   .append('text')
35.   .text(function (d) {
36.   return d;
37.   })
38.   .style('color', function (d) {
39.   if (d > 30) {
40.   return 'red';
41.   }
42.   })
43.   </script>
```

坐标轴

坐标轴，是可视化图表中经常出现的一种图形，由一些列线段和刻度组成。坐标轴在 SVG 中是没有现成的图形元素的，需要用其他的元素组合构成。D3 提供了坐标轴的组件，如此在 SVG 画布中绘制坐标轴变得像添加一个普通元素一样简单。

坐标轴由什么构成

坐标轴在可视化图形中是很重要的一部分，很多图表的展示都需要使用坐标轴，例如：柱形图、折线图。

D3中的坐标轴：

SVG 画布的预定义元素里，有六种基本图形：

- 矩形
- 圆形
- 椭圆
- 线段
- 折线
- 多边形

还有一种比较特殊的存在，也是最强的元素：

- 路径

所以说，在D3种是没有现成的坐标轴组件的，需要我们使用别的方式使用坐标轴。

我们可以使用类似下面的方式：

```
<g>
<!-- 第一个刻度 -->
<g>
<line></line> <!-- 第一个刻度的直线 -->
<text></text> <!-- 第一个刻度的文字 -->
</g>
<!-- 第二个刻度 -->
<g>
<line></line> <!-- 第二个刻度的直线 -->
<text></text> <!-- 第二个刻度的文字 -->
</g>
...
<!-- 坐标轴的轴线 -->
<path></path>
</g>
```

分组元素 ，是 SVG 画布中的元素，意思是 group。此元素是将其他元素进行组合的容器，在这里是用于将坐标轴的其他元素分组存放。

如果需要手动添加这些元素就太麻烦了，为此，D3 提供了一个组件：d3.axisBottom(xScale)。它为我们完成了以上工作。

使用坐标轴

定义坐标轴

坐标轴通常需要和比例尺一起使用：

```
// 为坐标轴定义一个线性比例尺
var xScale = d3.scaleLinear()
 .domain([0, d3.max(dataset)])
 .range([0, 250]);
// 定义一个坐标轴
var xAxis = d3.axisBottom(xScale) //定义一个axis，由bottom可知，是朝下的
 .ticks(7); //设置刻度数目
```

定义坐标轴相关的函数：

- d3.svgAxis()：D3 中坐标轴的组件，能够在 SVG 中生成组成坐标轴的元素。
- scale()：指定比例尺。
- orient()：指定刻度的朝向，bottom 表示在坐标轴的下方显示。
- ticks()：指定刻度的数量。

添加坐标轴

上面我们定义好了坐标轴，接下来就是将其添加到画布中去。

```
svg.append("g")
 .call(axis);
```

上面有一个 call() 函数，其参数是前面定义的坐标轴 axis。

设定坐标轴的样式和位置

默认的坐标轴样式不太美观，下面提供一个常见的样式：

```
<style>
.axis path,
.axis line{
 fill: none;
 stroke: black;
 shape-rendering: crispEdges;
}

.axis text {
 font-family: sans-serif;
 font-size: 11px;
}
</style>
```

分别定义了类 axis 下的 path、line、text 元素的样式。接下来，只需要将坐标轴的类设定为 axis 即可。
坐标轴的位置，可以通过 transform 属性来设定。
通常在添加元素的时候就一并设定，写成如下形式：

```
svg.append("g")
 .attr("class","axis")
 .attr("transform","translate(20,130)")
 .call(axis)
```

完整的实例

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

```
3.
4.  <head>
5.   <title>比例尺与坐标轴</title>
6.
7.   <script type="text/javascript" src="http://d3js.org/d3.v5.min.js">
8.   </script>
9.
10.  <meta name="keywords" content="keyword1,keyword2,keyword3">
11.  <meta name="description" content="this is my page">
12.  <meta name="content-type" content="text/html; charset=UTF-8">
13.
14.  <style>
15.  .axis path,
16.  .axis line {
17.  fill: none;
18.  stroke: black;
19.  shape-rendering: crispEdges;
20.  }
21.
22.  .axis text {
23.  font-family: sans-serif;
24.  font-size: 11px;
25.  }
26.  </style>
27.
28.  </head>
29.
30.  <body>
31.
32.  <svg width="960" height="600"></svg>
33.  <script>
34.  // 2、定义画布位置
35.  var marge = { top: 60, bottom: 60, left: 60, right: 60 }
36.
37.  // 4、定义比例尺，才能绘制彩条
38.  // 定义线性比例尺
39.  var dataset = [2.5, 2.1, 1.7, 1.3, 0.9];
40.  var scaleLinear = d3.scaleLinear()
41.  .domain([0, d3.max(dataset)])
42.  .range([0, 250]);
43.
```

```
44.  // 定义序列比例尺
45.  var index = [0, 1, 2, 3, 4];
46.  var color = ["red", "blue", "green", "yellow", "black"];
47.
48.  var ordinal = d3.scaleOrdinal()
49.  .domain(index)
50.  .range(color);
51.
52.  ordinal(0); //返回 red
53.  ordinal(2); //返回 green
54.  ordinal(4); //返回 black
55.
56.  // 1、线绘制画布
57.  var svg = d3.select("svg");
58.  var g = svg.append("g")
59.  .attr("transform", "translate(" + marge.top + "," + marge.left + ")");// 设置画布的位
置
60.
61.  // 3、定义矩形条的高度
62.  var rectHeight = 30;
63.
64.  g.selectAll("rect")
65.  .data(dataset)
66.  .enter()
67.  .append("rect")
68.  .attr("x", 20)
69.  .attr("y", function (d, i) {
70.  return i * rectHeight;
71.  })
72.  .attr("width", function (d) {
73.  return scaleLinear(d);
74.  })
75.  .attr("height", rectHeight - 5)
76.  .attr("fill", function (d) {
77.  return ordinal(d); // 这里使用比例尺，来为每个矩形填充颜色
78.  });
79.
80.  // 5、定义坐标轴
81.  // 为坐标轴定义一个线性比例尺
82.  var xScale = d3.scaleLinear()
83.  .domain([0, d3.max(dataset)])
```

```
    .range([0, 250])

    // 定义一个坐标轴
    var xAxis = d3.axisBottom(xScale)//定义一个axis，由bottom可知，是朝下的
    .ticks(7)//设置刻度数目

    g.append("g")
    .attr("class","axis") // 6、最后一步，为坐标轴定义样式
    .attr("transform", "translate(" + 20 + "," + (dataset.length * rectHeight) + ")")
    .call(xAxis)
    </script>
</body>
</html>
```

完整的柱形图

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>
    <script src="http://d3js.org/d3.v5.min.js"></script>
    <style>
    #app {
    margin: 0 auto;
    width: 500px;
    height: 400px;
    background: #efefef;
    position: relative;
    }

    .bar {
    /* width: 30px; */
    /* height: 50px; */
    fill: green;
    position: absolute;
    bottom: 100px;
    }
```

```
}

.bar span {
display: block;
text-align: center;
}
</style>
</head>

<body>
<svg id="app" style="width: 500px; height:400;"></svg>

<script>

const SVG_HEIGHT = 400;
const SVG_WIDTH = 500;
const MARGIN = { TOP: 30, RIGHT: 30, BOTTOM: 30, LEFT: 30 }

//模拟数据
var datalist = [20, 30, 40, 50, 15]

//容器（画布）
var container = d3.select("#app")

//线性比例尺
const yScale = d3.scaleLinear()
.domain([0, d3.max(datalist)])
.range([SVG_HEIGHT - MARGIN.TOP - MARGIN.BOTTOM, 0])

var axisLeft = d3.axisLeft(yScale)

//序数比例尺
var xScale = d3.scaleBand()
.domain(d3.range(datalist.length))
.range([0, SVG_WIDTH - MARGIN.LEFT - MARGIN.RIGHT])
.paddingInner(0.1)

//x轴比例尺展示无意义
var axisBottom = d3.axisBottom(xScale)

axisBottom(
```

```
container.append('g')
.attr('transform', `translate(${MARGIN.LEFT},${SVG_HEIGHT - MARGIN.TOP})`))

axisLeft(
container
.append('g')
.attr('transform', 'translate(30,30)')
)

container.selectAll('rect')
.data(datalist)
.enter()
.append('rect')
.classed('bar', true)

.attr('x', function (d, i) {
return xScale(i) + MARGIN.LEFT + 'px';
})
.attr('width', function (d, i) {
return xScale.bandwidth() + 'px';
})
.attr('y', function (d, i) {
return SVG_HEIGHT - MARGIN.TOP + 'px';
})
.attr('height', function () {
return 0
})
.transition()
.duration(1000)
.delay(function (d, i) {
return i * 200
})
.attr('y', function (d, i) {
return yScale(d) + MARGIN.TOP + 'px';
})
.style('height', function (d, i) {
return SVG_HEIGHT - MARGIN.TOP - MARGIN.BOTTOM - yScale(d) + 'px';
})

container.append('g').attr('class', 'textGrop')

```

```
  d3.select('.textGrop')
  .selectAll('text')
  .data(datalist)
  .enter()
  .append('text')
  .attr('text-anchor', 'middle')
  .text(function (d, i) {
  return d
  })
  .attr('x', function (d, i) {
  return xScale(i) + MARGIN.LEFT + xScale.bandwidth() / 2 + 'px'
  })
  .attr('y', function (d, i) {
  return SVG_HEIGHT - MARGIN.TOP - 10 + 'px'
  })
  .style('fill', function (d) {
  return 'green'
  })
  .transition()
  .duration(1000)
  .delay(function (d, i) {
  return i * 200
  })
  .attr('y', function (d, i) {
  return yScale(d) + MARGIN.TOP - 10 + 'px'
  })

  </script>
</body>
</html>
```

交互式操作

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>Document</title>
```

```
<script src="https://d3js.org/d3.v5.min.js"></script>
<style>
#app {
margin: 0 auto;
width: 600px;
height: 400px;
background: #efefef;
position: relative;
margin: auto 0;
}

#tooltip {
display: none;
background: #666;
color: white;
border-radius: 6px;
height: 50px;
width: 80px;
position: absolute;
left: 0px;
top: 0px;
z-index: 1;
transition: all 100ms;
}

.bar {
fill: rgb(104, 152, 241);
position: absolute;
bottom: 100px;
}

.bar span {
display: block;
text-align: center;
}

.tool {
text-align: center;
}
</style>
```

```
</head>

<body>
 <div id="tooltip">

 </div>
 <div style="text-align: center;">
 <svg id="app" style="width: 600px; height:400;"></svg>
 </div>
 <hr>
 <div class="tool">
 <button id="btn-sort">排序</button>
 <button id="btn-add">添加</button>
 <button id="btn-update">更新</button>
 </div>
 <script>

 const SVG_HEIGHT = 400;
 const SVG_WIDTH = 600;
 const MARGIN = { TOP: 30, RIGHT: 30, BOTTOM: 30, LEFT: 30 };

 //排序标记
 var sort_flag = false;

 //模拟数据
 var datalist = [20, 30, 40, 50, 15];

 //容器（画布）
 var container = d3.select("#app");

 //y轴线性比例尺
 var yScale = d3.scaleLinear()
 .domain([0, d3.max(datalist)])
 .range([SVG_HEIGHT - MARGIN.TOP - MARGIN.BOTTOM, 0]);
 var axisLeft = d3.axisLeft(yScale);

 //x轴，序数比例尺
 var xScale = d3.scaleBand()
 .domain(d3.range(datalist.length))
 .range([0, SVG_WIDTH - MARGIN.LEFT - MARGIN.RIGHT])
 .paddingInner(0.1);
```

```
90.
91.   //x轴比例尺展示无意义
92.   var axisBottom = d3.axisBottom(xScale);
93.   axisBottom(
94.   container.append('g')
95.   .attr('transform', `translate(${MARGIN.LEFT},${SVG_HEIGHT - MARGIN.TOP})`))
96.
97.   //添加左侧坐标轴
98.   axisLeft(
99.   container
100.  .append('g')
101.  .attr('transform', 'translate(30,30)')
102.  )
103.
104.
105.  function renderRect() {
106.
107.  //添加新的rect
108.  container.selectAll('rect')
109.  .data(datalist)
110.  .enter()
111.  .append('rect')
112.  .classed('bar', true)
113.  .on('click', function (d) {
114.  let x = d3.event.pageX;
115.  let y = d3.event.pageY;
116.  d3.select("#tooltip")
117.  .style('display', 'block')
118.  .style('left', x + 'px')
119.  .style('top', y + 'px')
120.  .text(function () {
121.  return d;
122.  })
123.  })
124.
125.  //更新样式
126.  container.selectAll('rect')
127.  .attr('x', function (d, i) {
128.  return xScale(i) + MARGIN.LEFT + 'px';
129.  })
130.  .attr('width', function (d, i) {
```

```
131.    return xScale.bandwidth() + 'px';
132.    })
133.    .style('height', function () {
134.    return '0px';
135.    })
136.    .attr('y', function (d, i) {
137.    return SVG_HEIGHT - MARGIN.TOP + 'px';
138.    })
139.    .on('mouseover', function () {
140.    d3.select(this).style('fill', 'orange');
141.    })
142.    .on('mouseout', function () {
143.    d3.select(this).style('fill', 'rgb(104, 152, 241)');
144.    })
145.    .transition()
146.    .duration(200)
147.    .delay(function (d, i) {
148.    return i * 100
149.    })
150.    .attr('y', function (d, i) {
151.    return yScale(d) + MARGIN.TOP + 'px';
152.    })
153.    .style('height', function (d, i) {
154.    return SVG_HEIGHT - MARGIN.TOP - MARGIN.BOTTOM - yScale(d) + 'px';
155.    })
156.
157.    }
158.
159.    function renderText() {
160.
161.    container.append('g')
162.    .attr('class', 'textGrop');
163.
164.    d3.select('.textGrop')
165.    .selectAll('text')
166.    .data(datalist)
167.    .enter()
168.    .append('text')
169.    .attr('text-anchor', 'middle')
170.    .text(function (d, i) {
171.    return d;
```

```
172.  });
173.
174.  d3.select('.textGrop')
175.  .selectAll('text')
176.  .attr('x', function (d, i) {
177.  return xScale(i) + MARGIN.LEFT + xScale.bandwidth() / 2 + 'px';
178.  })
179.  .attr('y', function (d, i) {
180.  return SVG_HEIGHT - MARGIN.TOP - MARGIN.BOTTOM - 5 + 'px';
181.  })
182.  // .style('fill', function (d) {
183.  // return 'red';
184.  // })
185.  .transition()
186.  .duration(200)
187.  .delay(function (d, i) {
188.  return i * 100;
189.  })
190.  .attr('y', function (d, i) {
191.  return yScale(d) + MARGIN.TOP - 5 + 'px';
192.  })
193.  }
194.
195.  //刷新比例尺（当数据有变化时需要执行）
196.  function refreshScale() {
197.  yScale.domain([0, d3.max(datalist)]);
198.  xScale.domain(d3.range(datalist.length))
199.  }
200.
201.  function sort() {
202.  container.selectAll('rect').sort((a, b) => {
203.  return sort_flag ? d3.descending(a, b) : d3.ascending(a, b);
204.  })
205.  .transition()
206.  .duration(500)
207.  .attr('x', (d, i) => {
208.  return xScale(i) + MARGIN.LEFT + 'px';
209.  })
210.
211.
212.  container.select('.textGrop')
```

```
212.    container.select('.textGroup')
213.    .selectAll('text').sort((a, b) => {
214.    return sort_flag ? d3.descending(a, b) : d3.ascending(a, b);
215.    })
216.    .text(function (d, i) {
217.    return d;
218.    })
219.    .transition()
220.    .duration(500)
221.    .attr('x', (d, i) => {
222.    return xScale(i) + MARGIN.LEFT + xScale.bandwidth() / 2 + 'px';
223.    })
224.
225.    sort_flag = !sort_flag;
226.
227.    }
228.
229.    function initEvent() {
230.    d3.select("#btn-sort").on('click', () => {
231.    sort();
232.    })
233.
234.    d3.select("#btn-add").on('click', () => {
235.    let num = Math.ceil(Math.random() * 100);
236.    datalist.push(num);
237.    refreshScale();
238.    renderRect();
239.    renderText();
240.    })
241.
242.
243.    d3.select("#btn-update").on('click', () => {
244.    mockData();
245.    refreshScale();
246.    renderRect();
247.    renderText();
248.    })
249.    }
250.
251.    function mockData() {
252.    datalist = [];
```

```
  for (let i = 0; i < 10; i++) {
  let num = Math.ceil(Math.random() * 100);
  this.datalist.push(num);
  }
  }

  renderRect();
  renderText();
  initEvent();

  </script>

</body>

</html>
```

饼状图

模拟数据 data.csv

```
education,population
大专以及以上,11964
高中和中专,18799
初中,51966
小学,35876
文盲人口,5466
```

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>Document</title>
  <script src="./d3.v5.min.js"></script>
</head>

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

  <script>
```

```
d3.csv("data.csv", function (d) {
return {
education: d.education,
population: +d.population,
}
}).then(data => {
console.log(data);
var sum = d3.sum(data.map(function (d) {
return d.population
}))

for (i in data) {
data[i].Percentage = (data[i].population / sum * 100).toFixed(0) + "%";
}
console.log(data);

var width = 800,
height = 800,
margin = { "left": 30, "top": 30, "right": 30, "bottom": 30 },
svg_width = width + margin.left + margin.right,
svg_height = height + margin.top + margin.bottom,
font_size = 15;

var svg = d3.select("#container")
.append("svg")
.attr("width", width)
.attr("height", height)


var Pie = svg.append("g")
.attr("transform", "translate(" + width / 2 + "," + height / 2 + ")")

var arc_generator = d3.arc()
.innerRadius(width / 8)
.outerRadius(width / 4)
// .startAngle(0)
// .endAngle(120*Math.PI/180);

var angle_data = d3.pie()
.value(function (d) {
return d.population;
```

```
57.   })
58.   console.log(angle_data(data));
59.
60.   var color = d3.schemeCategory10;
61.   console.log(color)
62.
63.   //生成内部圆环
64.   Pie.selectAll("path")
65.   .data(angle_data(data))
66.   .enter()
67.   .append("path")
68.   .attr("d", arc_generator)
69.   .style("fill", function (d, i) {
70.   return color[i];
71.   })
72.   .attr("class", ".path")
73.
74.   //标注
75.   var arc_label = d3.arc()
76.   .innerRadius(width / 4)
77.   .outerRadius(width / 2)
78.
79.   Pie.selectAll(".arc_label")
80.   .data(angle_data(data))
81.   .enter()
82.   .append("path")
83.   .attr("d", arc_label)
84.   .attr("class", ".arc_label")
85.   .style("fill", "none")
86.
87.   //画标注线
88.   function line_label(angle_data) {
89.   var str = ""
90.   var i = -0;
91.   for (d in angle_data) {
92.   str = "M" + arc_generator.centroid(angle_data[d])[0] + "," +
arc_generator.centroid(angle_data[d])[1];
93.   str = str + "L" + arc_label.centroid(angle_data[d])[0] + "," +
arc_label.centroid(angle_data[d])[1]
94.   // console.log(str);
95.   Pie.append("path")
```

```
  .attr("d", str)
  .attr("stroke", color[i])
  .attr("stroke-width", 2)
  i++;
  if (i > 10) i = 0;
  }
  }

  line_label(angle_data(data));

  var text = Pie.selectAll("text")
  .data(angle_data(data))
  .enter()
  .append("text")
  .attr("transform", function (d) {
  return "translate(" + arc_label.centroid(d) + ")"
  })
  .attr("text-anchor", function (d) {
  var x = arc_label.centroid(d)[0];
  return x <= 0 ? "end" : "start";
  })
  .attr("font-size", font_size)
  .style("fill", function (d, i) {
  return color[i];
  })
  .style("text-decoration", "underline")
  .text(function (d) {
  return d.data.education + d.data.Percentage;
  })
  })
  </script>
 </body>

 </html>
```

散点图

```
<!DOCTYPE html>
<html lang="en">

<head>
```

```
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>Document</title>
  <script src="./d3.v5.min.js"></script>
</head>

<body>
  <div id="app">


  </div>
  <hr>
  <button id="btn-update">update</button>


  <script>

  const svg_width = 600;
  const svg_height = 450;
  const padding = 30;
  const colors = d3.schemeCategory10;

  var container = d3.select("#app");
  //生成画布
  var svg = container.append('svg')
  .attr('width', svg_width)
  .attr('height', svg_height)
  .style('background', '#efefef');


  //生成模拟数据
  var dataset = [];
  function mockData() {
  dataset = [];
  for (let i = 0; i < 10; i++) {
  let x = Math.ceil(Math.random() * 400);
  let y = Math.ceil(Math.random() * 400);
  dataset.push({
  x,
```

```
    y
    })
    }
    }
    mockData();


    //添加散点
    function addCircle() {
    svg
    .append('g')
    .attr('class', 'grop_circle')
    .attr('clip-path', 'url(#rect-clip-path)')
    .selectAll('circle')
    .data(dataset)
    .enter()
    .append('circle')
    .attr('r', 15)
    .attr('cx', (d, i) => {
    return d.x;
    })
    .attr('cy', (d, i) => {
    return d.y;
    })
    .attr('fill', (d, i) => {
    return colors[i]
    })
    // .
    // on('click', function (d) {
    // alert(d.x);
    // })

    }
    addCircle();


    //添加坐标轴
    function addAxis() {
    //x轴的线性比例尺
    this.x_scale = d3.scaleLinear()
    .domain([0, d3.max(dataset, (d) => {
```

```
86.   return d.x;
87.   })])
88.   .range([0, svg_width - padding * 2]);
89.
90.   //y轴的线性比例尺
91.   this.y_scale = d3.scaleLinear()
92.   .domain([0, d3.max(dataset, (d) => {
93.   return d.y;
94.   })])
95.   .range([svg_height - padding * 2, 0]);
96.
97.   //x方向坐标轴
98.   this.x_axis = d3.axisBottom(x_scale)
99.   svg.append('g')
100.  .attr('id', 'x_axis')
101.  .call(this.x_axis)
102.  .attr('transform', `translate(${padding},${svg_height - padding})`);
103.
104.  //y方向坐标轴
105.  this.y_axis = d3.axisLeft(y_scale)
106.  svg.append('g')
107.  .attr('id', 'y_axis')
108.  .call(y_axis)
109.  .attr('transform', `translate(${padding},${padding})`);
110.
111.  }
112.  addAxis();
113.
114.  //添加裁切路径
115.  function addClipPath() {
116.  svg.
117.  append('clipPath')
118.  .attr('id', 'rect-clip-path')
119.  .append('rect')
120.  .attr('x', padding)
121.  .attr('y', padding)
122.  .attr('width', svg_width - padding * 2)
123.  .attr('height', svg_height - padding * 2);
124.
125.  }
126.  addClipPath();
```

```
127.
128.  //数据更新
129.  d3.select("#btn-update").on('click', () => {
130.  mockData();
131.
132.  //更新坐标轴
133.  x_scale.domain([0, d3.max(dataset, (d) => {
134.  return d.x;
135.  })])
136.
137.  y_scale.domain([0, d3.max(dataset, (d) => {
138.  return d.y;
139.  })])
140.
141.  this.x_axis = d3.axisBottom(x_scale)
142.  svg.select("#x_axis")
143.  .transition()
144.  .duration(500)
145.  .call(this.x_axis);
146.  svg.select("#y_axis")
147.  .transition()
148.  .duration(500)
149.  .call(this.y_axis);
150.
151.
152.  svg.select('.grop_circle')
153.  .selectAll('circle')
154.  .data(dataset)
155.  .transition()
156.  .duration(500)
157.  .attr('cx', (d, i) => {
158.  return d.x;
159.  })
160.  .attr('cy', (d, i) => {
161.  return d.y;
162.  })
163.  .attr('fill', (d, i) => {
164.  return colors[i]
165.  })
166.  })
167.
```

```
168.
169.   </script>
170.  </body>
171.  </html>
```

地图可视化

在数据可视化中，地图是很重要的一部分。很多情况会与地图有关联，如中国各省的人口多少，GDP多少等，都可以和地图联系在一起。

D3地图绘制
制作地图需要 JSON 文件，将 JSON 的格式应用于地理上的文件，叫做 GeoJSON 文件。

投影函数

```
1.  var projection = d3.geo.mercator()//投影函数
2.   .center([107, 31])//设定地图的中心位置--经度和纬度
3.   .scale(850)//设定放大的比例
4.   .translate([width/2, height/2]);//设定平移
```

由于 GeoJSON 文件中的地图数据，都是经度和纬度的信息。它们都是三维的，而要在网页上显示的是二维的，所以要设定一个投影函数来转换经度纬度。如上所示，使用 d3.geo.mercator() 的投影方式。

地理路径生成器
为了根据地图的地理数据生成 SVG 中 path 元素的路径值，需要用到 d3.geo.path()，称为地理路径生成器。

```
1.  var path = d3.geo.path()
2.   .projection(projection);
```

projection() 是设定生成器的投影函数，把上面定义的投影传入即可。

案例

index.html

```
1.  <!DOCTYPE html>
2.  <html lang="en">
3.
4.  <head>
5.   <meta charset="UTF-8">
6.   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7.   <meta http-equiv="X-UA-Compatible" content="ie=edge">
8.   <title>Document</title>
9.   <script src="http://d3js.org/d3.v5.min.js"></script>
```

```
<script src="http://d3js.org/d3.v3.min.js"></script>
<script src="./map.js"></script>
<style>
#tooltip {
text-align: center;
padding: 20px;
margin: 10px;
font: 12px sans-serif;
background: lightsteelblue;
border: 1px;
border-radius: 2px;
pointer-events: none;
position: absolute;
left: -20px;
top: -20px;
z-index: 1;
border: 1px solid grey;
border-radius: 5px;
font-size: 12px;
width: auto;
padding: 4px;
color: white;
}

#tooltip h4 {
margin: 0;
font-size: 14px;
}

#tooltip table {
table-layout: fixed;
}

#tooltip tr td {
padding: 0;
margin: 0;
}

#tooltip tr td:nth-child(1) {
width: 50px;
}
```

```
50.
51.  #tooltip tr td:nth-child(2) {
52.  text-align: center;
53.  }
54.  </style>
55. </head>
56.
57. <body>
58.  <div id="tooltip"></div>
59.  <div id="container"></div>
60.  <script>
61.
62.  function tooltipHtml(d) {
63.  return "<h4>" + d.properties.name + "</h4><table>" +
64.  "<tr><td>Low</td><td>" + (d.properties.adcode) + "</td></tr>" +
65.  "<tr><td>Average</td><td>" + (d.properties.center) + "</td></tr>" +
66.  "<tr><td>High</td><td>" + (d.properties.centroid) + "</td></tr>" +
67.  "</table>"
68.  }
69.
70.  d3.json('./china.json').then((result) => {
71.  // console.log(result);
72.  map('#container', result, function (d) {
73.  d3.select("#tooltip").html(
74.  tooltipHtml(d, event)
75.  )
76.  .style("left", (d3.event.pageX) + "px")
77.  .style("top", (d3.event.pageY - 28) + "px")
78.  })
79.  })
80.  </script>
81. </body>
82.
83. </html>
```

map.js

```
1.  function map(id, data, clickCbk) {
2.  const width = 1000;
3.  const height = 1000;
4.
5.  // 画布
```

```
const svg = d3
.select(id)
.append('svg')
.attr('width', width)
.attr('height', height)

//投影方式。
const projection = d3
//投影函数，地理投影，可将经纬度转成平面坐标系
.geoMercator()
//设定地图的中心位置--经度和纬度
.center([107, 31])
//设定放大的比例
.scale(800)
//设定平移
.translate([width / 2, height / 2]);

const path = d3.geoPath().projection(projection);
const color = d3.schemeCategory10;

svg
.selectAll('g')
.data(data.features)
.enter()
.append('g')
.append('path')
.attr('d', path)
.attr('stroke', '#000')
.attr('stroke-width', 1)
.attr('opacity', 0.6)
.attr('fill', function (d, i) {
return color[i % 10]
})
.on('click', function (d, i) {
console.log(d);
clickCbk(d, d3.event);
})
.on('mouseover', function () {
d3.select(this).attr('opacity', 1);
})
.on('mouseout', function () {
```

```
    .on('mouseout', function () {
      d3.select(this).attr('opacity', 0.6);
    });

    // 添加坐标
    svg
      .selectAll('g')
      .append('text')
      .attr('font-size', 12)
      .attr('text-anchor', 'middle')
      .attr('x', d => {
        const position = projection(d.properties.centroid || [0, 0]);
        return position[0];
      })
      .attr('y', d => {
        const position = projection(d.properties.centroid || [0, 0]);
        return position[1];
      })
      .attr('dy', d => {
        //这里为什么这么写呢，因为澳门和香港重合了，挤到一起了。
        if (d.properties.name === '澳门') {
          return 15;
        }
      })
      .text(d => d.properties.name);
  }
```

china.json

china.json 获取地址

```
{
  "type": "FeatureCollection",
  "features": [{
  "type": "Feature",
  "properties": {
  "adcode": 110000,
  "name": "北京市",
  "center": [116.405285, 39.904989],
  "centroid": [116.41989, 40.189913],
  "childrenNum": 16,
  "level": "province",
  "subFeatureIndex": 0
```

```
"subFeatureIndex": 0,
"acroutes": [100000],
"parent": {
"adcode": 100000
}
},
"geometry": {
"type": "MultiPolygon",
"coordinates": [
[
[
[117.210024, 40.082262],
[117.105315, 40.074479],
[117.105315, 40.074479],
[117.102851, 40.073563],
[117.102235, 40.073105],
[117.102235, 40.073105],
[117.102851, 40.073563],
[116.999989, 40.030053],
[116.927924, 40.054788],
[116.783794, 40.035093],
[116.757925, 39.968176],
[116.786874, 39.886963],
[116.926076, 39.835524],
[116.949482, 39.778529],
[116.902055, 39.763813],
[116.90575, 39.687883],
[116.812128, 39.616018],
[116.717273, 39.603572],
[116.717273, 39.603572],
[116.720969, 39.599884],
[116.720969, 39.599884],
[116.726512, 39.595274],
[116.726512, 39.595274],
[116.703106, 39.588819],
[116.703106, 39.588819],
[116.607636, 39.619705],
[116.524484, 39.596657],
[116.440716, 39.527466],
[116.433325, 39.44296],
[116.332927, 39.457744],
```

```
53.     [116.245464, 39.515466],
54.     [116.204196, 39.588819],
55.     [116.10195, 39.576368],
56.     [116.10195, 39.576368],
57.     [115.957204, 39.561147],
58.     [115.910393, 39.600345],
59.     [115.910393, 39.600345],
60.     [115.855574, 39.554689],
61.     [115.855574, 39.554689],
62.     [115.846951, 39.550999],
63.     [115.846951, 39.550999],
64.     [115.821081, 39.517312],
65.     [115.821081, 39.517312],
66.     [115.752712, 39.512696],
67.     [115.752712, 39.512696],
68.     [115.738545, 39.539464],
69.     [115.738545, 39.539925],
70.     [115.738545, 39.539464],
71.     [115.738545, 39.539925],
72.     [115.737314, 39.544078],
73.     [115.737314, 39.544078],
74.     [115.723763, 39.544539],
75.     [115.723763, 39.544539],
76.     [115.721299, 39.543617],
77.     [115.721299, 39.543617],
78.     [115.721299, 39.55146],
79.     [115.721299, 39.55146],
80.     [115.716988, 39.560225],
81.     [115.716988, 39.560225],
82.     [115.699125, 39.577751],
83.     [115.698509, 39.577751],
84.     [115.698509, 39.577751],
85.     [115.699125, 39.577751],
86.     [115.698509, 39.577751],
87.     [115.69543, 39.579135],
88.     [115.69543, 39.579135],
89.     [115.586408, 39.58928],
90.     [115.478619, 39.650578],
91.     [115.478619, 39.650578],
92.     [115.498945, 39.69617],
93.     [115.498945, 39.69617],
```

```
[115.443511, 39.785426],
[115.443511, 39.785426],
[115.567314, 39.816224],
[115.514344, 39.837821],
[115.522967, 39.898898],
[115.426264, 39.95029],
[115.454597, 40.029595],
[115.599343, 40.11979],
[115.73485, 40.129398],
[115.773038, 40.176044],
[115.85311, 40.148609],
[115.89869, 40.234536],
[115.968907, 40.264219],
[115.9184, 40.354103],
[115.861733, 40.364589],
[115.861733, 40.364589],
[115.779197, 40.442501],
[115.755792, 40.540333],
[115.907929, 40.617133],
[116.005247, 40.58397],
[116.088399, 40.62667],
[116.22021, 40.744181],
[116.247311, 40.791762],
[116.464738, 40.771827],
[116.334159, 40.90446],
[116.473977, 40.895867],
[116.455499, 40.98084],
[116.519557, 40.981292],
[116.519557, 40.981292],
[116.599013, 40.974516],
[116.615643, 41.053072],
[116.688324, 41.044499],
[116.677853, 40.970902],
[116.730208, 40.897676],
[116.858323, 40.833423],
[116.964881, 40.70972],
[117.110858, 40.70836],
[117.286401, 40.660719],
[117.386799, 40.684317],
[117.49582, 40.674334],
[117.389879, 40.5617],
```

```
    [117.344299, 40.582152],
    [117.213104, 40.512136],
    [117.225423, 40.369148],
    [117.309191, 40.279284],
    [117.309807, 40.279284],
    [117.309191, 40.279284],
    [117.309807, 40.279284],
    [117.389879, 40.228141],
    [117.367089, 40.172387],
    [117.367089, 40.172844],
    [117.367089, 40.173301],
    [117.367089, 40.173301],
    [117.367089, 40.172844],
    [117.367089, 40.172387],
    [117.344299, 40.13443],
    [117.210024, 40.082262]
   ]
   ]
   ]
   }
   }
   //...
  ]}
```

前言

> *本小册是《千锋大前端小册》系列之 大数据可视化 部分。内容包含E-Chars基础、E-Chars实战及D3基础和进阶。全部内容是依照《千锋教育大前端大纲-大数据可视化》编写。*
>
> *—— 作者：千锋教育·古艺散人*

公安警情可视化

本项目应用 Vue.js 和 D3.js 开发的一套公安警情可视化系统。

环境搭建

将结果数据的每一个数据项，作为单个图元元素展示，大量的数据集构成数据图像，同时将数据的各个属性

以多维度的方式展现，从而提高数据的可读性

```
vue create d3-project
yarn add d3 mockjs axios jquery -S
```

搭建页面结构

1、修改 main.js

```
// /main.js
import Vue from 'vue'
import App from './App.vue'

import './assets/css/reset.css'
import './assets/css/common.css'

import zoom from './assets/scripts/tool/zoom'
zoom()
window.addEventListener('resize', zoom)

Vue.config.productionTip = false

new Vue({
  render: h => h(App),
}).$mount('#app')
```

2、添加基本样式

```
/* /src/assets/css/reset.css */
html,
body,
div,
span,
applet,
object,
iframe,
h1,
h2,
h3,
h4,
h5
```

```
h3,
h6,
p,
blockquote,
pre,
a,
abbr,
acronym,
address,
big,
cite,
code,
del,
dfn,
em,
img,
ins,
kbd,
q,
s,
samp,
small,
strike,
strong,
sub,
sup,
tt,
var,
b,
u,
i,
center,
dl,
dt,
dd,
ol,
ul,
li,
fieldset,
form,
label,
```

```
legend,
table,
caption,
tbody,
tfoot,
thead,
tr,
th,
td,
article,
aside,
canvas,
details,
embed,
figure,
figcaption,
footer,
header,
hgroup,
menu,
nav,
output,
ruby,
section,
summary,
time,
mark,
audio,
video {
 margin: 0;
 padding: 0;
 border: 0;
 font-size: 100%;
 font: inherit;
 vertical-align: baseline;
}

article,
aside,
details,
figcaption,
```

```
figure,
footer,
header,
hgroup,
menu,
nav,
section {
 display: block;
}

body {
 line-height: 1;
}

li,
ol,
ul {
 list-style: none;
}

blockquote,
q {
 quotes: none;
}

blockquote:before,
blockquote:after,
q:before,
q:after {
 content: '';
 content: none;
}

table {
 border-collapse: collapse;
 border-spacing: 0;
}

a{
 text-decoration: none;
}
```

```
/* /src/assets/css/common.css */
body {
  display: flex;
  transform-origin: 0px 0px 0px;
  overflow: hidden;
  position: relative;
  font-family: "微软雅黑";
}

.text-white-shadow {
  text-shadow: 0 0 5px #fbfbfb, 0 0 10px #fbfbfb, 0 0 20px #228DFF, 0 0 35px #228DFF, 0 0 75px #228DFF;
}

.text-green-shadow {
  text-shadow: 0 0 5px #68fff3, 0 0 10px #68fff3, 0 0 20px #228DFF, 0 0 35px #228DFF, 0 0 75px #228DFF;
}

.chart-title {
  font-size: 52px;
  color: #fff;
  position: absolute;
  top: 10px;
  left: 50px;
}

.auto-tooltip {
  box-sizing: border-box;
  position: absolute;
  padding: 10px 15px;
  background: rgba(59, 57, 54, 0.8);
  border-radius: 5px;
  border: 1px solid #928a82;
  color: #fff;
  font-size: 30px;
  z-index: 9999;
  white-space: nowrap;
  display: none;
}

```

```
.axis path,
.axis line {
  fill: none;
  stroke: none;
  shape-rendering: optimizeSpeed;
}

.axis text {
  font-size: 28px;
  font-family: sans-serif;
  fill: #a4d5ff;
}

.legend-hide {
  color: #8da5e4 !important;
}

.map-chart .xingzheng {
  transform: translate(0px, 8px);
}

.map-chart .xingshi .circleMain {
  fill: #0084ff;
}

.map-chart .xingzheng .circleMain {
  fill: #fff;
}

.map-chart .xingshi .circleOuter1,
.xingshi .circleOuter2 {
  stroke: #0084ff;
  fill: none;
}

.map-chart .xingzheng .circleOuter1,
.xingzheng .circleOuter2 {
  stroke: #fff;
  fill: none;
}
```

```
80.
81. .circleOuter2 {
82.  animation: cri 1s infinite;
83. }
84.
85. @keyframes cri {
86.  0%,
87.  100%,
88.  20%,
89.  40%,
90.  60%,
91.  80% {
92.  -webkit-transition-timing-function: cubic-bezier(0.215, .61, .355, 1);
93.  transition-timing-function: cubic-bezier(0.215, .61, .355, 1)
94.  }
95.  0% {
96.  transform: scale(1);
97.  }
98.  100% {
99.  transform: scale(1.5);
100.  opacity: 0;
101.  }
102. }
103.
104. @-webkit-keyframes cri {
105.  0%,
106.  100%,
107.  20%,
108.  40%,
109.  60%,
110.  80% {
111.  -webkit-transition-timing-function: cubic-bezier(0.215, .61, .355, 1);
112.  transition-timing-function: cubic-bezier(0.215, .61, .355, 1)
113.  }
114.  0% {
115.  -webkit-transform: scale(1);
116.  }
117.  100% {
118.  -webkit-transform: scale(1.5);
119.  -webkit-opacity: 0;
120.  }
```

```
}

/* 添加页面背景图 */
.side-r-wrap {
  flex: 1;
  height: 100%;
  background: url(../../assets/images/common/page-bg.jpg) no-repeat;
  background-size: 100% 100%;
  position: relative;
  overflow: hidden;
}
```

3、实现页面自动缩放

```
// /main.js
import zoom from './assets/scripts/tool/zoom'
zoom()
window.addEventListener('resize', zoom)
// /src/assets/scripts/tool/zoom.js
import config from './config'

export default () => {
  const {
  pageWidth,
  pageHeight
  } = config

  const body = document.querySelector('body')

  body.style.width = `${pageWidth}px`
  body.style.height = `${pageHeight}px`
  const x = window.innerWidth / pageWidth
  const y = window.innerHeight / pageHeight
  body.style.transform = `scale(${x}, ${y})`
}
// /src/assets/scripts/tool/config.js
export default {
  pageWidth: 3360,
  pageHeight: 1890
}
```

实现顶部信息展示

1、App 组件 修改

/App.vue

```
<template>
 <div class="castle side-r-wrap">
 <TopSide :title="title"></TopSide>
 <TopMid></TopMid>
 </div>
</template>

<script>
import TopSide from '@/components/TopSide'
import TopMid from '@/components/TopMid'

export default {
 name: 'Castle',
 data () {
 return {
 title: '公安警情可视化'
 }
 },
 components: {
 TopSide,
 TopMid
 }
}
</script>
<style scoped>
</style>
```

2、构建TopSide.vue

/src/components/TopSide.vue

Mock模块已经安装

```
<template>
 <div class="top-side">
 <div class="title">{{title}}</div>
```

```
  <div class="title">{{title}}</div>
  <div class="time">{{yyyyMMdd}}
  <strong>{{HHmm}}</strong>
  </div>
  </div>
</template>
<script>
  import Mock from 'mockjs'
  export default {
  props: ['title'],
  name: 'topSide',
  data () {
  return {
  yyyyMMdd: '',
  HHmm: ''
  }
  },
  mounted () {
  const date = Mock.Random.now('yyyyMMddHHmm')
  const year = date.substring(0, 4)
  const month = date.substring(4, 6)
  const day = date.substring(6, 8)
  const hour = date.substring(8, 10)
  const minute = date.substring(10, 12)
  this.yyyyMMdd = `${year}.${month}.${day}`
  this.HHmm = `${hour}:${minute}`
  }
  }
</script>
<style>
  .top-side {
  position: absolute;
  top: 0;
  left: 0;
  width: 100%;
  }

  .title {
  font-size: 58px;
  color: #fdfeff;
  position: absolute;
```

```
  left: 62px;
  top: 30px;
  text-shadow: 0 0 5px #fbfbfb, 0 0 10px #fbfbfb, 0 0 20px #228DFF, 0 0 35px #228DFF, 0 0 75px #228DFF;
  }

  .time {
  position: absolute;
  top: 30px;
  right: 55px;
  color: #fdfeff;
  font-size: 36px;
  text-shadow: 0 0 5px #fbfbfb, 0 0 10px #fbfbfb, 0 0 20px #228DFF, 0 0 35px #228DFF, 0 0 75px #228DFF;
  }

  .time strong {
  font-size: 60px;
  }

 </style>
```

3、构建 TopMid.vue

/src/components/TopSide.vue

```
 <template>
  <div class="top-mid">
  <div class="item" v-for="item in dataset" :key="item.id">
  <span class="name">{{item.name}}
  <strong>{{item.value}}</strong>
  </span>
  <span class="huanbi">环比
  <strong :data-state="item.state">{{item.huanbi}}
  <em>%</em>
  </strong>
  </span>
  </div>
  </div>
 </template>
 <script>
  import axios from 'axios'
```

```
import api from '../assets/scripts/tool/api'
import Data from '../data/fantasy/castle/top'
Data()
export default {
name: 'topMid',
data () {
return {
dataset: []
}
},
mounted () {
const self = this
axios.get(api.castleTop)
.then(response => {
const data = response.data.result.top
data.map(item => {
let state
let huanbi
if (item.huanbi > 0) {
state = 'up'
huanbi = item.huanbi
} else if (item.huanbi === 0) {
state = 'level'
huanbi = item.huanbi
} else if (item.huanbi < 0) {
state = 'down'
huanbi = Math.abs(item.huanbi)
}
self.dataset.push({
name: item.name,
value: item.value,
huanbi: huanbi,
state: state
})
})
})
.catch(error => {
console.error(error)
})
}
```

```
  }
</script>
<style>
 .top-mid {
 position: absolute;
 left: 50%;
 top: 30px;
 width: 1650px;
 display: flex;
 justify-content: space-around;
 box-sizing: border-box;
 padding: 24px 20px 0;
 background: url(../assets/images/common/top-center-bg.png) no-repeat top center;
 background-size: 100% 100%;
 transform: translate(-50%, 0);
 margin-left: -20px;
 }

 .item {
 max-width: 33.33%;
 overflow: hidden;
 white-space: nowrap;
 }

 .name {
 font-size: 40px;
 color: #b4c7f9;
 }

 .name strong {
 font-size: 50px;
 color: #ff8244;
 font-weight: normal;
 margin: 0 10px;
 }

 .huanbi {
 font-size: 30px;
 color: #b4c7f9;
 }

```

```
.huanbi strong {
font-size: 40px;
margin: 0 10px 0 36px;
position: relative;
display: inline-block;
}

.huanbi em {
font-size: 30px;
}

.huanbi strong[data-state]:after {
content: "";
display: inline-block;
position: absolute;
width: 25px;
height: 26px;
top: 14px;
left: -30px;
}

.huanbi strong[data-state="up"] {
color: #ff4444;
}

.huanbi strong[data-state="level"] {
color: #b4c7f9;
}

.huanbi strong[data-state="down"] {
color: #44ff86;
}

.huanbi strong[data-state="up"]:after {
background: url(../assets/images/common/huanbi-up.png) no-repeat;
}

.huanbi strong[data-state="down"]:after {
background: url(../assets/images/common/huanbi-down.png) no-repeat;
}

```

```
</style>
```

/src/assets/scripts/tool/api

```
const isOnline = false
const onlineApiHost = isOnline ? 'http://88.888.88.88:8888/project/' : 'http://localhost:8080/'

export default {
  castleTop: onlineApiHost + 'fantasy/castle/top',
  iotalarm: onlineApiHost + 'iot/overview/alarm',
  iottop5: onlineApiHost + 'iot/overview/top5',
  iottrend: onlineApiHost + 'iot/overview/trend',
  district: onlineApiHost + 'stride/fireworks/city',
  list: onlineApiHost + 'stride/fireworks/list',
  dotemap: onlineApiHost + 'fantasy/castle/dotemap'
}
```

/src/data/fantasy/castle/top

```
import {
  urlReg
} from '../../../assets/scripts/tool/utils'

import Mock from 'mockjs'

const data = () => {
  Mock.mock(urlReg('fantasy/castle/top'), {
  'code': 1,
  'msg': 'success',
  'result': {
  'top|3': [{
  'name|+1': ['本周', '本月', '本年'],
  'value': '@natural(100,10000)',
  'huanbi': '@integer(-100,100)'
  }]
  }
  })
}

export default data
```

/src/assets/scripts/tool/utils

```
import $ from 'jquery'

export const urlReg = (name) => {
  const protocols = '((https?|s?ftp|irc[6s]?|git|afp|telnet|smb):\\/\\/)?'
  const userInfo = '([a-z0-9]\\w*(\\:[\\S]+)?\\@)?'
  const domain = '([a-z0-9]([\\w]*[a-z0-9])*\\.)?[a-z0-9]\\w*\\.[a-z]{2,}(\\.[a-z]{2,})?'
  const port = '(:\\d{1,5})?'
  const ip = '\\d{1,3}\\.\\d{1,3}\\.\\d{1,3}\\.\\d{1,3}'
  const address = '(\\/\\S*)?'
  const domainType = [protocols, userInfo, domain, port, address, name, address]
  const ipType = [protocols, userInfo, ip, port, address, name, address]
  return new RegExp('^' + domainType.join('') + '$', 'i') || new RegExp('^' + ipType.join('') + '$', 'i')
}

export const tooltip = (option) => {
  const el = option.el
  let location = option.location
  const data = option.data
  const length = data.length
  let text = ''
  for (let i = 0; i < length; i++) {
  if (data[i].color) {
  text += `<span style="color:${data[i].color}">${data[i].name} : ${data[i].value}</span><br>`
  } else {
  text += `<span>${data[i].name} : ${data[i].value}</span><br>`
  }
  }
  $(el).html(text)
  const globalWidth = $('body').outerWidth()
  const elWidth = $(el).outerWidth()
  const elHeight = $(el).outerHeight()
  location.x = location.x - elWidth / 2
  location.y = location.y - elHeight - 10
  if (location.x + elWidth / 2 > globalWidth) {
  location.x = globalWidth - elWidth
  }
  $(el).css({
```

```
  left: location.x,
  top: location.y
  })
  return $(el)
 }
```

构建左侧信息展示

1、App 组件 修改

/App.vue

```
<template>
 <div class="castle side-r-wrap">
 <TopSide :title="title"></TopSide>
 <TopMid></TopMid>
 <Left></Left>
 </div>
</template>

<script>
import TopSide from '@/components/TopSide'
import TopMid from '@/components/TopMid'
import Left from '@/components/Left'

export default {
 name: 'Castle',
 data () {
 return {
 title: '公安警情可视化'
 }
 },
 components: {
 TopSide,
 TopMid,
 Left
 }
}
</script>
<style scoped>
```

```
</style>
```

2、构建 Left.vue

/src/components/Left.vue

```
<template>
 <div class="left">
 <div class="diandongche">
 <h2 class="chart-title">年度刑事案件数
 </h2>
 <ul class="alarm-list">
 <li>
 <p class="alarm-list-name">
 <span>今年</span>
 </p>
 <p class="alarm-list-value">{{thisyear}}</p>
 </li>
 <li>
 <p class="alarm-list-name">
 <span>环比</span>
 </p>
 <p class="alarm-list-value" :data-state="hbstate">{{huanbi}}</p>
 </li>
 <li>
 <p class="alarm-list-name">
 <span>去年</span>
 </p>
 <p class="alarm-list-value">{{lastyear}}</p>
 </li>
 </ul>
 </div>
 <div class="caseResolved">
 <h2 class="chart-title">年度出警数</h2>
 <div class="case-resolved-chart" id="caseResolvedChart"></div>
 </div>
 </div>
</template>
<script>
 import axios from 'axios'
 import api from '../assets/scripts/tool/api'
 import Data from '../data/fantasy/castle/left'
```

```
import Data from '../data/fantasy/castle/left'
import MultiPie from '../assets/scripts/charts/multiPie'
Data()

export default {
name: 'left',
data () {
return {
thisyear: '',
hbstate: '',
huanbi: '',
lastyear: ''
}
},
mounted () {
const self = this
axios.get(api.iotalarm)
.then(response => {
const data = response.data.result
self.dealDDC(data.diandongche)
self.dealCase(data.caseResolved)
})
.catch(error => {
console.error(error)
})
},
methods: {
dealDDC (data) {
let hbstate
let hbValue = data.huanbi
if (hbValue < 0) {
hbstate = 'down'
hbValue = Math.abs(hbValue)
} else if (hbValue === 0) {
hbstate = 'level'
hbValue = '- 0'
} else if (hbValue > 0) {
hbstate = 'up'
hbValue = Math.abs(hbValue)
}
this.thisyear = data.thisyear
```

```
77.   this.hbstate = hbstate
78.   this.huanbi = hbValue
79.   this.lastyear = data.lastyear
80.   },
81.   dealCase (data) {
82.   const config = {}
83.   const multiPie = new MultiPie('.case-resolved-chart', config)
84.   multiPie.render(data)
85.   }
86.   }
87.   }
88. </script>
89. <style scoped>
90.   .diandongche {
91.   position: absolute;
92.   top: 220px;
93.   left: 60px;
94.   width: 680px;
95.   height: 600px;
96.   background: url(../assets/images/common/tip-title-bg.png) no-repeat top left;
97.   }
98.
99.   .alarm-list {
100.  position: absolute;
101.  top: 120px;
102.  left: 0px;
103.  width: 100%;
104.  height: 100%;
105.  font-size: 0;
106.  }
107.
108.  .alarm-list li {
109.  width: 50%;
110.  height: 190px;
111.  overflow: hidden;
112.  display: inline-block;
113.  margin-bottom: 10px;
114.  }
115.
116.  .alarm-list-name span {
117.  font-size: 50px;
```

```
118.  color: #b4c7f9;
119.  position: relative;
120.  display: inline-block;
121.  }
122.
123.  .alarm-list-name span:after {
124.  content: "件";
125.  display: inline-block;
126.  position: absolute;
127.  top: 4px;
128.  right: -80px;
129.  text-align: center;
130.  font-size: 30px;
131.  line-height: 46px;
132.  color: #8da5e4;
133.  height: 46px;
134.  width: 60px;
135.  background: #0c3f87;
136.  border: 1px solid #443cba;
137.  }
138.
139.  .alarm-list li:nth-child(2) .alarm-list-name span:after {
140.  display: none;
141.  }
142.
143.  .alarm-list-value {
144.  font-size: 60px;
145.  color: #1aac4e;
146.  position: relative;
147.  display: inline-block;
148.  margin-top:20px;
149.  }
150.
151.  .alarm-list li:first-child .alarm-list-value {
152.  font-size: 90px;
153.  color: #44ff86;
154.  }
155.
156.  .alarm-list li:nth-child(2) .alarm-list-value {
157.  margin-top: 34px;
158.  text-indent: 40px;
```

```
}

.alarm-list li:nth-child(2) .alarm-list-value[data-state="up"] {
color: #ff4444;
}

.alarm-list li:nth-child(2) .alarm-list-value[data-state="level"] {
color: #b4c7f9;
}

.alarm-list li:nth-child(2) .alarm-list-value[data-state="down"] {
color: #44ff86;
}

.alarm-list li:nth-child(2) .alarm-list-value:after {
content: "%";
display: inline-block;
position: absolute;
bottom: 4px;
right: -24px;
font-size: 30px;
}

.alarm-list li:nth-child(2) .alarm-list-value[data-state]:before {
content: "";
display: inline-block;
position: absolute;
width: 25px;
height: 26px;
top: 26px;
left: 0px;
}

.alarm-list li:nth-child(2) .alarm-list-value[data-state="up"]:before {
background: url(../assets/images/common/huanbi-up.png) no-repeat;
}

.alarm-list li:nth-child(2) .alarm-list-value[data-state="down"]:before {
background: url(../assets/images/common/huanbi-down.png) no-repeat;
}
```

```
199.
200.  .caseResolved {
201.  position: absolute;
202.  top: 830px;
203.  left: 60px;
204.  background: url(../assets/images/common/tip-title-bg.png) no-repeat top left;
205.  }
206.
207.  .case-resolved-chart {
208.  width: 900px;
209.  height: 270px;
210.  margin-top: 130px;
211.  }
212. </style>
```

/src/data/fantasy/castle/left.js

```
1.  import {
2.  urlReg
3.  } from '../../../assets/scripts/tool/utils'
4.
5.  import Mock from 'mockjs'
6.
7.  const data = () => {
8.  Mock.mock(urlReg('/iot/overview/alarm'), {
9.  'code': 1,
10.  'msg': 'success',
11.  'result': {
12.  'diandongche': {
13.  'lastyear': '@natural(1,2000)',
14.  'thisyear': '@natural(1,2000)',
15.  'huanbi': '@integer(-100,100)'
16.  },
17.  'caseResolved|3': [{
18.  'name|+1': ['部门一', '部门二', '部门三'],
19.  'value': '@natural(1,2000)'
20.  }]
21.  }
22.  })
23.  }
24.  export default data
```

/src/assets/scripts/charts/multiPie.js

/src/assets/scripts/charts/multiPie.js

```
import * as d3 from 'd3'

export default class MultiPie {
 /**
 * 默认配置项
 * @return {[Object]} [默认配置项]
 */
 defaultSetting() {
 return {
 width: 900,
 height: 270,
 radius: [50, 66], // [innerRadius,outerRadius]
 gap: 100, // 相邻两个图形的间距
 margin: { // 多个图形布局：从左往右，竖直方向按容器高度居中放置，故只设置左侧距离left即可
 left: 10
 },
 label: { // 名称文本样式
 normal: {
 fontSize: 32,
 color: '#46aaff',
 anchor: 'middle',
 cursor: 'pointer',
 top: 46 // 名称文本距离图案顶部的距离
 },
 emphasis: {
 fontSize: 32,
 color: '#74ffd3',
 anchor: 'middle',
 cursor: 'pointer',
 top: 46 // 名称文本距离图案顶部的距离
 }
 },
 itemStyle: {
 label: { // value值文本样式
 fontSize: 32,
 color: '#46aaff',
 anchor: 'middle',
 cursor: 'pointer',
 top: 10 // value值文本距离容器中线的偏移距离，默认放在饼图正中间
```

```
    top: 10 // value值表示距离容器下边的偏移距离，默认放在饼图垂直中间
  },
  color: [ // 饼图填充色
  ['#4a8ce5', 'black'],
  ['#44ff86', 'black'],
  ['#dccc5c', 'black']
  ]
  }
  }
  }
  /**
  * 初始化，创建容器
  * @param {String} selector 图表容器，支持class或id
  * @param {Object} option 配置项，控制图形样式
  * @return {[type]} [description]
  */
  constructor(selector, option = {}) {
  const defaultSetting = this.defaultSetting()
  this.config = Object.assign(defaultSetting, option)
  const {
  width,
  height
  } = this.config
  // 创建svg
  this.svg = d3.select(selector)
  .append('svg')
  .attr('width', width)
  .attr('height', height)
  }
  /**
  * 处理原始数据，获取pie布局转换后的数据
  * @param {Array} data 原始数据
  * @return {Array} dataset 转换后的数据
  */
  getDataset(data) {
  let dataset = []
  const clockwisePie = d3.pie() // 顺时针，针对数据类型:[small,bigger]
  const anticlockwisePie = d3.pie() // 逆时针,针对数据类型：[bigger,small]
  .startAngle(0)
  .endAngle(-2 * Math.PI)
  // 求取总数：sum
```

```
let sum = 0
data.map(d => {
sum += parseInt(d.value, 10)
})
data.map((d) => {
let value = d.value
let rate = Math.max(Math.floor(value * 100 / sum), 1)
let rateData = [rate, 100 - rate]
let dealData = rate >= 50 ? clockwisePie(rateData) : anticlockwisePie(rateData)
dataset.push(dealData)
})
return dataset
}
/**
* 绘制图案底部的名称文本
* @param {Object} chart 包裹文本的外层g容器
* @param {Object} info 单组原始数据，包括name和value
* @return {[type]} [description]
*/
renderName(chart, info) {
const {
radius: [, outerRadius],
label: {
normal: {
fontSize: fontSizeNor,
color: colorNor,
anchor: anchorNor,
top: topNor,
cursor: cursorNor
},
emphasis: {
fontSize: fontSizeEmp,
color: colorEmp,
anchor: anchorEmp,
top: topEmp,
cursor: cursorEmp
}
} = this.config
chart.select('.pie-name')
.attr('font-size', fontSizeNor)
```

```
121.   .attr('fill', colorNor)
122.   .attr('text-anchor', anchorNor)
123.   .attr('transform', `translate(0, ${outerRadius + topNor})`)
124.   .attr('cursor', cursorNor)
125.   .text(info.name)
126.   .on('mouseover', function () {
127.   d3.select(this)
128.   .attr('font-size', fontSizeEmp)
129.   .attr('fill', colorEmp)
130.   .attr('text-anchor', anchorEmp)
131.   .attr('transform', `translate(0, ${outerRadius + topEmp})`)
132.   .attr('cursor', cursorEmp)
133.   })
134.   .on('mouseout', function () {
135.   d3.select(this)
136.   .attr('font-size', fontSizeNor)
137.   .attr('fill', colorNor)
138.   .attr('text-anchor', anchorNor)
139.   .attr('transform', `translate(0, ${outerRadius + topNor})`)
140.   .attr('cursor', cursorNor)
141.   })
142.   }
143.   /**
144.   * 绘制图案中间的value值文本
145.   * @param {Object} chart 包裹文本的外层g容器
146.   * @param {[type]} info 单组原始数据，包括name和value
147.   * @return {[type]} [description]
148.   */
149.   renderValue(chart, info) {
150.   const {
151.   itemStyle: {
152.   label: {
153.   fontSize,
154.   color,
155.   anchor,
156.   cursor,
157.   top
158.   }
159.   }
160.   } = this.config
161.   chart.select('.pie-value')
```

```
.attr('font-size', fontSize)
.attr('fill', color)
.attr('text-anchor', anchor)
.attr('transform', `translate(0,${top})`)
.attr('cursor', cursor)
.text(info.value)
}
/**
* 绘制单个Pie图案
* @param {Objec} chartName 单个图案的外层g容器
* @param {Array} pieData 绘制饼图的数据（已经过布局处理）
* @param {Object} info 该图案的原始数据，包括name和value
* @param {Array} color 填充饼图的两个颜色值
* @return {[type]} [description]
*/
creatPie(chartName, pieData, info, color) {
const {
radius: [innerRadius, outerRadius]
} = this.config
const arc = d3.arc()
.innerRadius(innerRadius)
.outerRadius(outerRadius)
const chart = this.svg.select(chartName)
const update = chart.selectAll('path').data(pieData)
const enter = update.enter()
update.exit().remove()
// 绘制饼图图案
enter.append('path')
chart.selectAll('path').data(pieData)
.attr('fill', (d, i) => {
return color[i]
})
.attr('d', d => {
return arc(d)
})
// 绘制名称--name
enter.append('text').attr('class', 'pie-name')
this.renderName(chart, info)

// 绘制value值
enter.append('text').attr('class', 'pie-value')
```

```
203.   this.renderValue(chart, info)
204.   }
205.   render(data) {
206.   let dataset = this.getDataset(data)
207.   const update = this.svg.selectAll('.item')
208.   .data(dataset)
209.   update.enter().append('g').attr('class', 'item')
210.   update.exit().remove()
211.   // 多个图形布局：从左往右，相邻图形间隔为配置项----config.gap
212.   const {
213.   height,
214.   radius: [, R],
215.   gap,
216.   margin: {
217.   left
218.   },
219.   itemStyle: {
220.   color
221.   }
222.   } = this.config
223.   this.svg.selectAll('.item').data(dataset)
224.   .attr('transform', (d, i) => {
225.   return `translate(${R + left + 2 * R * i + i * gap},${height / 2})`
226.   })
227.   .attr('class', (d, i) => {
228.   return `item${i} item`
229.   })
230.   // 逐个绘制饼图
231.   dataset.map((d, i) => {
232.   this.creatPie(`.item${i}`, d, data[i], color[i])
233.   })
234.   }
235.  }
```

绘制环状图

利用D3按照部门构造环状图，构建年底出警数可视化。

1、修改 Left.vue

模板

```
<div class="caseResolved">
 <h2 class="chart-title">年度出警数</h2>
 <div class="case-resolved-chart" id="caseResolvedChart"></div>
</div>
```

逻辑

```
mounted() {
 axios.get(api.castleLeft)
 .then(result => {
 // ...

 // 1、在mounted里添加：
 this.dealCase(data.caseResolved)
 })
},

// 2、在 methods 里添加：
dealCase (data) {
 const config = {}
 const multiPie = new MultiPie('.case-resolved-chart', config)
 multiPie.render(data)
}

// 3、导入模块
import MultiPie from '../assets/scripts/charts/multiPie'
```

2、安装 d3.js

```
yarn add d3 -S
```

3、构建 D3 环图

在 /src/assets/scripts/charts/ 下创建 multiPie.js 文件：

```
import * as d3 from 'd3'

export default class MultiPie {
 /**
```

```
5.    * 默认配置项
6.    * @return {[Object]} [默认配置项]
7.    */
8.    defaultSetting() {
9.    return {
10.   width: 900,
11.   height: 270,
12.   radius: [50, 66], // [innerRadius,outerRadius]
13.   gap: 100, // 相邻两个图形的间距
14.   margin: { // 多个图形布局：从左往右，竖直方向按容器高度居中放置，故只设置左侧距离left即可
15.   left: 10
16.   },
17.   label: { // 名称文本样式
18.   normal: {
19.   fontSize: 32,
20.   color: '#46aaff',
21.   anchor: 'middle',
22.   cursor: 'pointer',
23.   top: 46 // 名称文本距离图案顶部的距离
24.   },
25.   emphasis: {
26.   fontSize: 32,
27.   color: '#74ffd3',
28.   anchor: 'middle',
29.   cursor: 'pointer',
30.   top: 46 // 名称文本距离图案顶部的距离
31.   }
32.   },
33.   itemStyle: {
34.   label: { // value值文本样式
35.   fontSize: 32,
36.   color: '#46aaff',
37.   anchor: 'middle',
38.   cursor: 'pointer',
39.   top: 10 // value值文本距离容器中线的偏移距离，默认放在饼图正中间
40.   },
41.   color: [ // 饼图填充色
42.   ['#4a8ce5', 'black'],
43.   ['#44ff86', 'black'],
44.   ['#dccc5c', 'black']
```

```
45.   ]
46.   }
47.   }
48.   }
49.
50.   /**
51.   * 初始化，创建容器
52.   * @param {String} selector 图表容器，支持class或id
53.   * @param {Object} option 配置项，控制图形样式
54.   * @return {[type]} [description]
55.   */
56.   constructor(selector, option = {}) {
57.   const defaultSetting = this.defaultSetting()
58.   this.config = Object.assign(defaultSetting, option)
59.   const {
60.   width,
61.   height
62.   } = this.config
63.   // 创建svg
64.   this.svg = d3.select(selector)
65.   .append('svg')
66.   .attr('width', width)
67.   .attr('height', height)
68.   }
69.
70.   /**
71.   * 处理原始数据，获取pie布局转换后的数据
72.   * @param {Array} data 原始数据
73.   * @return {Array} dataset 转换后的数据
74.   */
75.
76.   getDataset(data) {
77.   let dataset = []
78.   const clockwisePie = d3.pie() // 顺时针，针对数据类型:[small,bigger]
79.   const anticlockwisePie = d3.pie() // 逆时针,针对数据类型：[bigger,small]
80.   .startAngle(0)
81.   .endAngle(-2 * Math.PI)
82.   // 求取总数：sum
83.   let sum = 0
84.   data.map(d => {
85.   sum += parseInt(d.value, 10)
```

```
86.   })
87.   data.map((d) => {
88.   let value = d.value
89.   let rate = Math.max(Math.floor(value * 100 / sum), 1)
90.   let rateData = [rate, 100 - rate]
91.   let dealData = rate >= 50 ? clockwisePie(rateData) : anticlockwisePie(rateData)
92.   dataset.push(dealData)
93.   })
94.   return dataset
95.   }
96.
97.   /**
98.   * 绘制图案底部的名称文本
99.   * @param {Object} chart 包裹文本的外层g容器
100.  * @param {Object} info 单组原始数据，包括name和value
101.  * @return {[type]} [description]
102.  */
103.  renderName(chart, info) {
104.  const {
105.  radius: [, outerRadius],
106.  label: {
107.  normal: {
108.  fontSize: fontSizeNor,
109.  color: colorNor,
110.  anchor: anchorNor,
111.  top: topNor,
112.  cursor: cursorNor
113.  },
114.  emphasis: {
115.  fontSize: fontSizeEmp,
116.  color: colorEmp,
117.  anchor: anchorEmp,
118.  top: topEmp,
119.  cursor: cursorEmp
120.  }
121.  }
122.  } = this.config
123.  chart.select('.pie-name')
124.  .attr('font-size', fontSizeNor)
125.  .attr('fill', colorNor)
126.  .attr('text-anchor', anchorNor)
```

```
.attr('text-anchor', anchorNor)
.attr('transform', `translate(0, ${outerRadius + topNor})`)
.attr('cursor', cursorNor)
.text(info.name)
.on('mouseover', function () {
d3.select(this)
.attr('font-size', fontSizeEmp)
.attr('fill', colorEmp)
.attr('text-anchor', anchorEmp)
.attr('transform', `translate(0, ${outerRadius + topEmp})`)
.attr('cursor', cursorEmp)
})
.on('mouseout', function () {
d3.select(this)
.attr('font-size', fontSizeNor)
.attr('fill', colorNor)
.attr('text-anchor', anchorNor)
.attr('transform', `translate(0, ${outerRadius + topNor})`)
.attr('cursor', cursorNor)
})
}

/**
* 绘制图案中间的value值文本
* @param {Object} chart 包裹文本的外层g容器
* @param {[type]} info 单组原始数据，包括name和value
* @return {[type]} [description]
*/
renderValue(chart, info) {
const {
itemStyle: {
label: {
fontSize,
color,
anchor,
cursor,
top
}
}
} = this.config
chart.select('.pie-value')
```

```
.attr('font-size', fontSize)
.attr('fill', color)
.attr('text-anchor', anchor)
.attr('transform', `translate(0,${top})`)
.attr('cursor', cursor)
.text(info.value)
}

/**
* 绘制单个Pie图案
* @param {Objec} chartName 单个图案的外层g容器
* @param {Array} pieData 绘制饼图的数据（已经过布局处理）
* @param {Object} info 该图案的原始数据，包括name和value
* @param {Array} color 填充饼图的两个颜色值
* @return {[type]} [description]
*/
creatPie(chartName, pieData, info, color) {
const {
radius: [innerRadius, outerRadius]
} = this.config
const arc = d3.arc()
.innerRadius(innerRadius)
.outerRadius(outerRadius)
const chart = this.svg.select(chartName)
const update = chart.selectAll('path').data(pieData)
const enter = update.enter()
update.exit().remove()
// 绘制饼图图案
enter.append('path')
chart.selectAll('path').data(pieData)
.attr('fill', (d, i) => {
return color[i]
})
.attr('d', d => {
return arc(d)
})
// 绘制名称--name
enter.append('text').attr('class', 'pie-name')
this.renderName(chart, info)

// 绘制value值
```

```
    enter.append('text').attr('class', 'pie-value')
    this.renderValue(chart, info)
    }

    render(data) {
    let dataset = this.getDataset(data)
    const update = this.svg.selectAll('.item')
    .data(dataset)
    update.enter().append('g').attr('class', 'item')
    update.exit().remove()
    // 多个图形布局：从左往右，相邻图形间隔为配置项----config.gap
    const {
    height,
    radius: [, R],
    gap,
    margin: {
    left
    },
    itemStyle: {
    color
    }
    } = this.config
    this.svg.selectAll('.item').data(dataset)
    .attr('transform', (d, i) => {
    return `translate(${R + left + 2 * R * i + i * gap},${height / 2})`
    })
    .attr('class', (d, i) => {
    return `item${i} item`
    })
    // 逐个绘制饼图
    dataset.map((d, i) => {
    this.creatPie(`.item${i}`, d, data[i], color[i])
    })
    }
}
```

绘制水泡图

D3绘制右侧水泡图。

1、修改 App.vue

添加 Right 组件。

```
<template>
 <div class="side-r-wrap">
 // ...
 <Right></Right>
 </div>
</template>

<script>
// ...
import Right from '@/components/Right'

export default {
 // ...

 components: {
 // ...
 Right
 }
}
</script>
```

2、构建 Right 组件

在 /src/components/ 创建 Right.vue：

```
<template>
 <div class="right">
 <div class="people-sex">
 <h2 class="chart-title">男女干警比例</h2>
 <div class="sex-chart" id="sexChart">
 <svg>
 <defs>
 <linearGradient id="outline" x1="0%" y1="0%" x2="0%" y2="100%">
 <stop offset="0%" style="stop-color: rgb(6, 124, 255); stop-opacity: 1;"></stop>
 <stop offset="100%" style="stop-color: rgb(160, 60, 218);stop-opacity: 1;"></stop>
 </linearGradient>
 <linearGradient id="innerBall" x1="0%" y1="0%" x2="0%" y2="100%">
```

```
<stop offset="0%" style="stop-color: rgb(6, 124, 255); stop-opacity: 1;"></stop>
<stop offset="100%" style="stop-color: rgb(160, 60, 218);stop-opacity: 1;"></stop>
</linearGradient>
</defs>
</svg>
</div>
<div class="sex-legend">
<p class="male">男性
<span>{{male}}</span>
<em>%</em>
</p>
<p class="female">女性
<span>{{100 - male}}</span>
<em>%</em>
</p>
</div>
</div>
</div>
</template>
<script>
import * as d3 from 'd3'
import axios from 'axios'
import WaterBall from '../assets/scripts/charts/waterBall'
import api from '../assets/scripts/tool/api'
import Data from '../data/bandTop'
Data()
export default {
name: 'right',
data () {
return {
male: 0
}
},
mounted () {
const self = this
axios.get(api.iottop5)
.then(response => {
const data = response.data.result
self.deal(data.sex)
})
.catch(error => {
```

```
.catch(error => {
console.error(error)
})
},
methods: {
deal (data) {
this.male = data.male
const config = {}
const waterBall = new WaterBall('#sexChart', config)
waterBall.drawCharts(data)
}
}
}
</script>
<style scoped>
.people-sex {
position: absolute;
top: 220px;
right: 70px;
width: 540px;
height: 516px;
background: url(../assets/images/common/tip-title-bg.png) no-repeat top left;
}

.sex-chart {
margin-top:70px;
}

.sex-legend {
position: absolute;
top: 50%;
right: 4%;
transform: translateY(-50%);
}

.sex-legend p {
font-size: 38px;
color: #fff;
padding-left: 40px;
line-height: 1.5;
}
```

```
94.
95.   .sex-legend p span {
96.   color: #44ff86;
97.   font-size: 40px;
98.   margin: 0 12px 0 8px;
99.   }
100.
101.  .sex-legend p em {
102.  color: #44ff86;
103.  font-size: 28px;
104.  }
105.
106.  .sex-legend .male {
107.  background: url(../assets/images/fantasy/castle/male-legend.png) no-repeat left
      center;
108.  }
109.
110.  .sex-legend .female {
111.  background: url(../assets/images/fantasy/castle/female-legend.png) no-repeat left
      center;
112.  }
113.
114.  .jizhan {
115.  position: absolute;
116.  top: 743px;
117.  right: 70px;
118.  width: 540px;
119.  height: 690px;
120.  background: url(../assets/images/common/tip-title-bg.png) no-repeat top left;
121.  }
122.
123.  .jizhan-list {
124.  margin-top: 136px;
125.  }
126.
127.  .jizhan-item {
128.  width: 100%;
129.  height: 100px;
130.  margin-top: 10px;
131.  background: url(../assets/images/fantasy/castle/top-item-bg.png) 0% 0% / 100% 100%
      no-repeat;
```

```
132.   }
133.
134.   .jizhan-item-index {
135.   float: left;
136.   height: 100%;
137.   line-height: 100px;
138.   width: 82px;
139.   text-align: center;
140.   color: #14c7fb;
141.   font-size: 36px;
142.   }
143.
144.   .jizhan-item-container {
145.   float: left;
146.   box-sizing: border-box;
147.   height: 100%;
148.   padding: 20px 0px 20px 10px;
149.   }
150.
151.   .jizhan-bar-container {
152.   width: 445px;
153.   height: 25px;
154.   line-height: 25px;
155.   }
156.
157.   .jizhan-back-bar {
158.   position: relative;
159.   float: left;
160.   width: 360px;
161.   height: 24px;
162.   margin-right: 10px;
163.   background: url(../assets/images/fantasy/castle/top-progress-bg.png) 0% 0% / 100% 100%
no-repeat;
164.   }
165.
166.   .jizhan-outer-bar {
167.   position: absolute;
168.   top: 0;
169.   left: 0;
170.   height: 24px;
171.   background: linear-gradient(to right, #1963a7 0%, #bec374 100%)
```

```
  }

  .jizhan-value {
  float: left;
  color: #3da3ff;
  font-size: 30px;
  }

  .jizhan-item-name {
  font-size: 30px;
  color: #b0caf9;
  }

 </style>
```

3、构建 api

修改 /src/assets/scripts/tool/api.js：

```
export default {
 // ...
 iottop5: onlineApiHost + 'iot/overview/top5'
}
```

4、制作模拟数据

在 /src/data 下创建 bandTop.js 文件：

```
import {
 urlReg
} from '../assets/scripts/tool/utils'

import Mock from 'mockjs'

const data = () => {
 Mock.mock(urlReg('/iot/overview/top5'), {
 'code': 1,
 'msg': 'success',
 'result': {
 'sex': {
 'male': '@natural(20,80)',
```

```
  'female': '@natural(20,80)'
  }
  }
  })
 }
 export default data
```

5、利用D3制作水泡图

在 /assets/scripts/charts/ 创建 waterBall.js 文件：

```
// /assets/scripts/charts/waterBall.js

/*
 * @Description: 水球图
 */
import * as d3 from 'd3'
export default class WaterBall {
 defaultSetting () {
 return {
 width: 400,
 height: 400,
 radius: 120,
 fillOuterLine: 'url(#outline)', // 外圈圆
 innerBall: 'url(#innerBall)' // 100% 实心圆填充
 }
 }

 constructor (selector, option) {
 const defaultSetting = this.defaultSetting()
 this.config = Object.assign(defaultSetting, option)
 const {
 width,
 height
 } = this.config

 // 创建svg
 this.svg = d3.select(selector).select('svg')
 .attr('width', width)
 .attr('height', height)
 this.defs = this.svg.select('defs')
```

```
  }

  drawCharts (data) {
  const {
  width,
  height,
  radius,
  innerBall,
  fillOuterLine
  } = this.config

  let rate = data.male
  const dataset = [
  [rate, 100 - rate]
  ]

  // 设置布局
  const clockwisePie = d3.pie() // 顺时针，针对数据类型:[small,bigger]
  .startAngle(Math.PI)
  .endAngle(3 * Math.PI)

  const anticlockwisePie = d3.pie() // 逆时针,针对数据类型：[bigger,small]
  .startAngle(Math.PI)
  .endAngle(-Math.PI)

  // 绘制男性
  const maleArc = d3.arc()
  .innerRadius(radius - 8)
  .outerRadius(radius + 8)

  // 绘制女性
  const femaleArc = d3.arc()
  .innerRadius(radius - 30)
  .outerRadius(radius - 14)

  // 处理好结构
  const ballUpdate = this.svg.selectAll('.ball')
  .data(dataset)

  const ballEnter = ballUpdate.enter().append('g').attr('class', 'ball')
  ballUpdate.exit().remove()
```

```
72.
73.   const ballGroup = this.svg.selectAll('.ball').data(dataset)
74.   .attr('transform', `translate(${width / 2 - 50},${height / 2})`)
75.
76.   // 绘制内部实心圆
77.   ballEnter.append('circle').attr('class', 'innerCircle')
78.   ballGroup.select('.innerCircle')
79.   .attr('r', radius - 14)
80.   .attr('fill', 'rgba(79,35,129,0.6)')
81.
82.   // 绘制100%的实心圆
83.   ballEnter.append('circle').attr('class', 'fillCircle')
84.   ballGroup.select('.fillCircle')
85.   .attr('r', radius - 14)
86.   .attr('fill', innerBall)
87.   .attr('clip-path', 'url(#areaWave)')
88.
89.   // 绘制外圈渐变填充圆
90.   ballEnter.append('circle').attr('class', 'outLine')
91.   ballGroup.select('.outLine')
92.   .attr('r', radius)
93.   .attr('fill', 'none')
94.   .attr('stroke', fillOuterLine)
95.   .attr('stroke-width', 4)
96.
97.   // 绘制外圈纯色填充圆 -- 男性占比
98.   ballEnter.append('path').attr('class', 'maleCircle')
99.   ballGroup.select('.maleCircle')
100.  .attr('fill', '#01e2fa')
101.  .attr('d', d => {
102.  return d[0] >= d[1] ? maleArc(clockwisePie(d)[0]) : maleArc(anticlockwisePie(d)[0])
103.  })
104.
105.  // 绘制外圈纯色填充圆 -- 女性占比
106.  ballEnter.append('path').attr('class', 'femaleCircle')
107.  ballGroup.select('.femaleCircle')
108.  .attr('fill', '#ff03b9')
109.  .attr('d', d => {
110.  const femaleData = [
111.  [d[1], d[0]]
112.  ]
```

```
113.  if (femaleData[0][0] >= femaleData[0][1]) {
114.  return femaleArc(clockwisePie(femaleData[0])[0])
115.  } else {
116.  return femaleArc(anticlockwisePie(femaleData[0])[0])
117.  }
118.  })
119.
120.  // 制作波浪纹 - clipPath
121.  const clipPathUpdate = this.defs.selectAll('clipPath').data(d3.range(1))
122.  clipPathUpdate.enter().append('clipPath').append('path')
123.  clipPathUpdate.exit().remove()
124.
125.  const waveClipCount = 2
126.  const waveClipWidth = radius * 4
127.  const waveHeight = 10.26
128.  const waveOffset = 0
129.  const waveCount = 1
130.
131.  let wavaData = []
132.  for (let i = 0; i <= 40 * waveClipCount; i++) {
133.  wavaData.push({
134.  x: i / (40 * waveClipCount),
135.  y: (i / (40))
136.  })
137.  }
138.
139.  const waveScaleX = d3.scaleLinear()
140.  .range([0, waveClipWidth])
141.  .domain([0, 1])
142.  const waveScaleY = d3.scaleLinear()
143.  .range([0, waveHeight])
144.  .domain([0, 1])
145.
146.  // translateY为radius 对应 0%
147.  // translateY为-radius 对应 100%
148.  const wavePercentScale = d3.scaleLinear()
149.  .domain([0, 100])
150.  .range([radius, -radius])
151.
152.  const clipArea = d3.area()
```

```
    .x(d => {
    return waveScaleX(d.x)
    })
    .y0(d => {
    return waveScaleY(Math.sin(Math.PI * 2 * waveOffset * -1 + Math.PI * 2 * (1 -
waveCount) + d.y * 2 * Math.PI))
    })
    .y1(2 * radius)

    let clipPath = this.defs.selectAll('clipPath')
    .attr('id', 'areaWave')
    .select('path')
    .datum(wavaData)
    .attr('d', clipArea)
    .attr('fill', 'yellow')

    clipPath.transition()
    .duration(2000)
    .attr('transform', `translate(${-3 * radius},${wavePercentScale(rate)})`)
    .on('start', () => {
    clipPath.attr('transform', `translate(${-3 * radius},${radius})`)
    })

    // 绘制图表名字
    ballEnter.append('text').attr('class', 'chart-name')
    ballGroup.select('.chart-name')
    .attr('y', -radius / 4)
    .attr('text-anchor', 'middle')
    .attr('fill', '#f4f8fc')
    .attr('font-weight', 'bold')
    .attr('font-size', 30)
    .text('男性占比')

    // 绘制百分占比数值 -- 严格的绘制顺序决定层级
    ballEnter.append('text').attr('class', 'valueText')
    ballGroup.select('.valueText')
    .attr('y', radius / 4 + 20)
    .attr('text-anchor', 'middle')
    .attr('fill', '#f4f8fc')
    .attr('font-size', 70)
    .text(0)
```

```
193.   .transition()
194.   .duration(3000)
195.   .on('start', function () {
196.   d3.active(this)
197.   .tween('text', function (d) {
198.   const that = d3.select(this)
199.   return function (t) {
200.   that.text(Math.floor(t * d[0]))
201.   }
202.   })
203.   })
204.
205.   // 绘制value值百分比符号
206.   ballEnter.append('text').attr('class', 'percentText')
207.   ballGroup.select('.percentText')
208.   .attr('y', 40)
209.   .attr('x', 70)
210.   .attr('text-anchor', 'middle')
211.   .attr('fill', '#fff')
212.   .attr('font-size', 40)
213.   .text('%')
214.
215.   // 用定时器做波浪动画
216.   setTimeout(function () {
217.   let distance = -3 * radius
218.   d3.timer(() => {
219.   distance++
220.   if (distance > -radius) {
221.   distance = -3 * radius
222.   }
223.   clipPath.attr('transform', `translate(${distance},${wavePercentScale(rate)})`)
224.   })
225.   }, 2000)
226.   }
227.  }
```