Echarts5.x

1.课程介绍与目标

有句话说的好“一图胜千言”，在我们开发的领域就是说，在对于复杂难懂且体量庞大的数据展示上面而言，图表的信息量要大得多，这也是我们为什么要谈数据可视化。

2.数据可视化介绍

数据可视化这一概念自1987年正式提出，经过30余年的发展，逐渐形成3个分支：科学计算可视化(scientific visualization)、信息可视化(information visualization)和可视分析(visual analytics)。近些年来，这3个子领域出现了逐渐融合的趋势。我们统称为“数据可视化”。

什么是数据可视化？

顾名思义，数据可视化就是将数据转换成图或表等，以一种更直观的方式展现和呈现数据。通过“可视化”的方式，我们看不懂的数据通过图形化的手段进行有效地表达，准确高效、简洁全面地传递某种信息，甚至我们帮助发现某种规律和特征，挖掘数据背后的价值。

同时关于数据可视化的定义有很多，像百度百科的定义是：数据可视化，是关于数据视觉表现形式的科学技术研究。其中，这种数据的视觉表现形式被定义为，一种以某种概要形式抽提出来的信息，包括相应信息单位的各种属性和变量。这种定义可能显得比较晦涩难懂。在大数据分析工具和软件中提到的数据可视化，就是利用运用计算机图形学、图像、人机交互等技术，将采集或模拟的数据映射为可识别的图形、图像。

数据可视化的展现形式

数据可视化有众多展现方式，不同的数据类型要选择适合的展现方法。在数据可视化中 除了常用的的柱状图、线状图、条形图、面积图、饼图、点图、仪表盘、走势图外，还有和弦图、圈饼图、金字塔、漏斗图、K线图、关系图、网络图、玫瑰图、帕累托图、数学公式图、预测曲线图、正态分布图、迷你图、行政地图、GIS地图等各种展现形式。都可以为我们提供丰富的图表选择，让我们在实际使用过程中有更好的展现方式。

我们可以通过类柱状图

比较类图表显示值与值之间的不同和相似之处。 使用图形的长度、宽度、位置、面积、角度和颜色来比较数值的大小， 通常用于展示不同分类间的数值对比，不同时间点的数据对比。

柱形图有别于直方图，柱状图无法显示数据在一个区间内的连续变化趋势。柱状图描述的是分类数据，回答的是每一个分类中"有多少？"这个问题。 需要注意的是，当柱状图显示的分类很多时会导致分类名重叠等显示问题。

同时可以通过占比类图表显示同一维度上的占比关系。饼图广泛应用在各个领域，用于表示不同分类的占比情况，通过弧度大小来对比各个分类。

饼图通过将一个圆饼按照分类的占比划分成多个区块，整个圆饼代表数据的总量，每个区块（圆弧）表示该分类占总体的比例大小，所有区块（圆弧）的加和等于 100%。

也可以趋势类折线图

趋势类图表显示数据的变化趋势。 使用图形的位置表现数据在连续区域上的分布，通常展示数据在连续区域上的大小变化的规律。

折线图用于显示数据在一个连续的时间间隔或者时间跨度上的变化，它的特点是反映事物随时间或有序类别而变化的趋势。

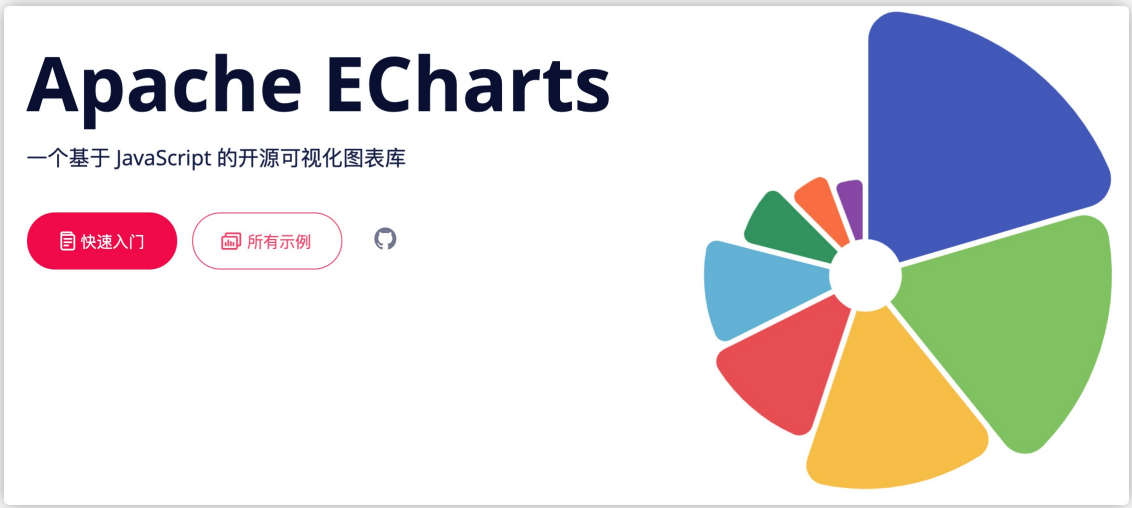
当然，大数据可视化的图表远远不止以上几种，最关键的是如何利用好这些工具及图表，归纳起来，一名数据可视化工程师需要具备三个方面的能力，数据分析能力、交互视觉能力、研发能力。

数据可视化有什么用？

数据可视化的意义是帮助人更好的分析数据，信息的质量很大程度上依赖于其表达方式。对数字罗列所组成的数据中所包含的意义进行分析，使分析结果可视化。其实数据可视化的本质就是视觉对话。数据可视化将技术与艺术完美结合，借助图形化的手段，清晰有效地传达与沟通信息。一方面，数据赋予可视化以价值；另一方面，可视化增加数据的灵性，两者相辅相成，帮助企业从信息中提取知识、从知识中收获价值。精心设计的图形不仅可以提供信息，还可以通过强大的呈现方式增强信息的影响力，吸引人们的注意力并使其保持兴趣，这是表格或电子表格无法做到的。

3.Echarts--商业级数据图表介绍

1.什么是Echarts



Echarts--商业级数据图表，它是一个纯JavaScript的图标库，可以流畅的运行在PC和移动设备上，兼容当前绝大部分浏览器（IE6/7/8/9/10/11，chrome，firefox，Safari等，底层依赖轻量级的Canvas类库ZRender，提供直观，生动，可交互，可高度个性化定制的数据可视化图表。创新的拖拽重计算、数据视图、值域漫游等特性大大增强了用户体验，赋予了用户对数据进行挖掘、整合的能力。

2. Echarts特点

1、丰富的可视化类型: 提供了常规的折线图、柱状图、散点图、饼图、K线图，用于统计的盒形图，用于地理数据可视化的地图、热力图、线图，用于关系数据可视化的关系图、treemap、旭日图，多维数据可视化的平行坐标，还有用于 BI 的漏斗图，仪表盘，并且支持图与图之间的混搭。

2、多种数据格式无需转换直接使用: 内置的 dataset 属性（4.0+）支持直接传入包括二维表，key-value 等多种格式的数据源，此外还支持输入 TypedArray 格式的数据。

3、**千万数据的前端展现**: 通过增量渲染技术（4.0+），配合各种细致的优化，ECharts 能够展现千万级的数据量。

4、**移动端优化**: 针对移动端交互做了细致的优化，例如移动端小屏上适于用手指在坐标系中进行缩放、平移。 PC 端也可以用鼠标在图中进行缩放（用鼠标滚轮）、平移等。

5、**多渲染方案，跨平台使用**: 支持以 Canvas、SVG（4.0+）、VML 的形式渲染图表。

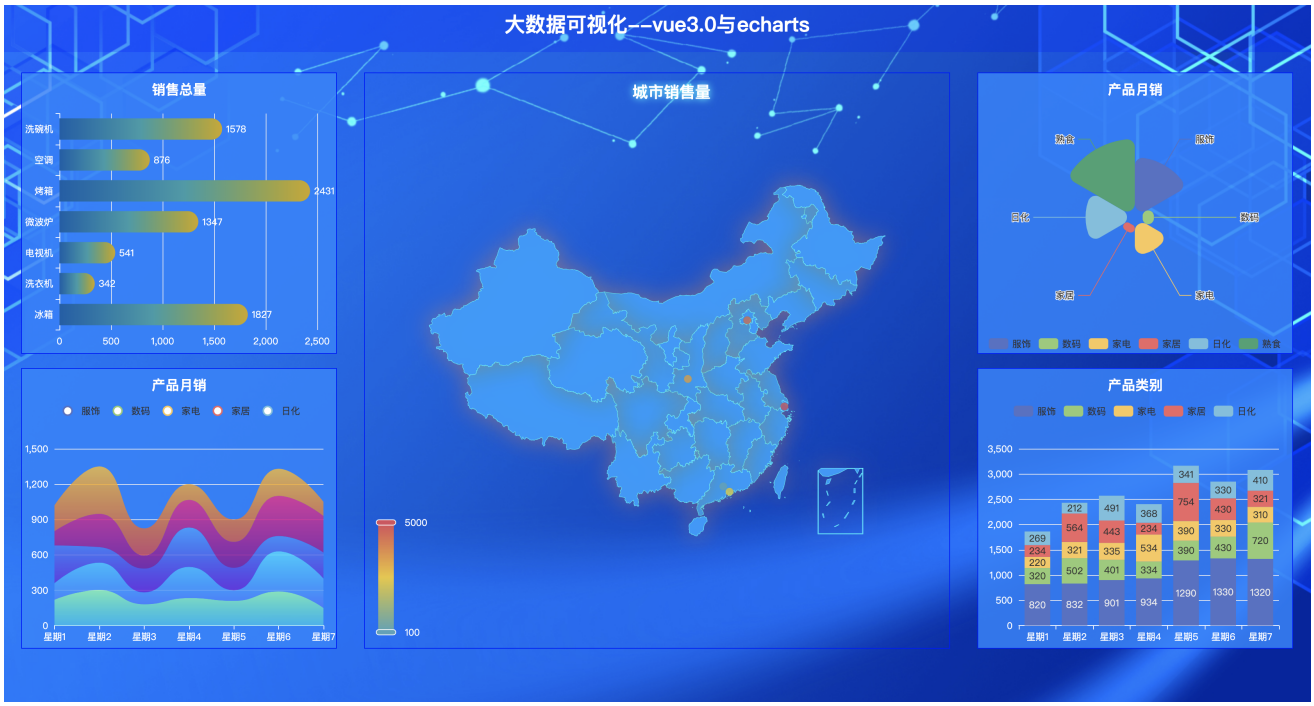
6、**深度的交互式数据探索**: 提供了 图例、视觉映射、数据区域缩放、tooltip、数据刷选等开箱即用的交互组件，可以对数据进行多维度数据筛取、视图缩放、展示细节等交互操作。

7、**多维数据的支持以及丰富的视觉编码手段**: 对于传统的散点图等，传入的数据也可以是多个维度的。

8、**动态数据**: 数据的改变驱动图表展现的改变。

9、**绚丽的特效**: 针对线数据，点数据等地理数据的可视化提供了吸引眼球的特效。

4. 项目演示



1.项目功能演示

2.项目启动与依赖安装

此过程会有引导 引导观众去获取资料

5.HelloWord 初体验

1.echarts获取

前期准备

电脑上面安装node

node下载地址： http://nodejs.cn/

淘宝镜像（选按）

淘宝 NPM 镜像站自 2014 年 正式对外服务。于npm命令在国内下载速度很慢。所以淘宝每隔10分钟就会把npm服务器的内容拉取一次放在国内服务器 这样一来我们在下载依赖的时候 速度会快很多

```
1    npm install -g cnpm --registry=https://registry.npm.taobao.org
2
```

安装

1.初始化 npm init -y

2.安装echarts依赖

npm install --save echarts

2.HelloWord

```
<template>
  <div class="about">
    <h1>This is an about page</h1>
    <div id="main"></div>
  </div>
</template>
<script>
import * as echarts from "echarts";
export default {
  mounted() {
    // echarts仅有一个方法init，执行init时传入一个具备大小
    // （如果没有指定容器的大小将会按照0大小来处理即无法看到图表）的dom节点后即可实例化出图表对象，图表库实现为多实例的，
    // 同一页面可多次init出多个图表。
    var myChart = echarts.init(document.getElementById("main")); // 绘制图表 // setOption方法设置图表实例的配置项 以及数据 所有参数和数据的修改都可以通过 setOption 完成，ECharts 会合并新的参数和数据，然后刷新图表。
    myChart.setOption({
      title: {
        //echarts标题
        text: "ECharts 入门示例",
      }, //tooltip:提示框组件，用于配置鼠标滑过或点击图表时的显示框。
      tooltip: {},
      // 不过我们在使用Echarts过程中经常会遇到如下问题：图例经常不知道如何调节到我们想要的位置。
      legend: {}, //横坐标 xAxis配置 直角坐标系x轴
      xAxis: {
        data: ["衬衫", "羊毛衫", "雪纺衫", "裤子", "高跟鞋", "袜子"],
      },
      yAxis: {}, //系列（series）
      // 系列（series）是很常见的名词。在 echarts 里，系列（series）是指：一组数值以及他们映射成的图。“系列”这个词原本可能来源于“一系列的数据”，而在 echarts 中取其扩展的概念，不仅表示数据，也表示数据映射成为的图
      // charts 里系列类型（series.type）就是图表类型。系列类型（series.type）至少有：line（折线图）、bar（柱状图）、pie（饼图）、scatter（散点图）、graph（关系图）、tree（树图）
      series: [
        {
          name: "销量",
          type: "bar",
          data: [5, 20, 36, 10, 10, 20],
        },
      ],
    });
  },
};
</script>
<style>
  #main{
    width: 500px;
    height: 500px;
  }

```

```
</style>
```

6. 配置项--title配置

title 标题组件，包含主标题和副标题。

```
<template>
  <div class="about">
    <h1>This is an about page</h1>
    <div id="main"></div>
  </div>
</template>
<script>
import * as echarts from "echarts";
export default {
  mounted() {
    var myChart = echarts.init(document.getElementById("main"));
    myChart.setOption({
      title: {
        show: true, //显示策略，默认值true,可选为：true（显示） | false（隐藏）
        text: "1主标题", //主标题文本，'\n'指定换行
        // link:'http://www.baidu.com', //主标题文本超链接,默认值true
        // target: "self", //指定窗口打开主标题超链接，支持'self' | 'blank'，不指定等同为'blank'（新窗口）
        subtext: '副标题', //副标题文本，'\n'指定换行
        // sublink: '', //副标题文本超链接
        // subtarget: null, //指定窗口打开副标题超链接，支持'self' | 'blank'，不指定等同为'blank'（新窗口）
        // x:'center', //水平安放位置，默认为'left'，可选为：'center' | 'left' | 'right' | {number}（x坐标，单位px）
        // y: 'bottom', //垂直安放位置，默认为top，可选为：'top' | 'bottom' | 'center' | {number}（y坐标，单位px）
        // backgroundColor: 'red', //标题背景颜色，默认'rgba(0,0,0,0)'透明
        // borderWidth: 5, //标题边框线宽，单位px，默认为0（无边框）
        // borderColor: '#ccffee', //标题边框颜色,默认'#ccc'
        // padding: 5, //标题内边距，单位px，默认各方向内边距为5，接受数组分别设定上右下左边距
        // itemGap: 10, //主副标题纵向间隔，单位px，默认为10
        // textStyle: { //主标题文本样式{"fontSize": 18,"fontWeight": "bolder","color": "#333"}
            // fontFamily: 'Arial, Verdana, sans...',
            // fontSize: 12,
            // fontStyle: 'normal',
            // fontWeight: 'normal', // },
            // subtextStyle: {//副标题文本样式{"color": "#aaa"}
            // fontFamily: 'Arial, Verdana, sans...',
            // fontSize: 12,
            // fontStyle: 'normal',
            // fontWeight: 'normal',
          // },
          // subtextStyle: {
          //   color: "#a1b2c3", // 副标题文字的颜色。
          //   fontStyle: "normal", // 副标题文字字体的风格。 'normal'  'italic'  'oblique'
```

```
            //   fontWeight: "bold", // 副标题文字字体的粗细。 'normal' 'bold'  'bolder'
'lighter' 500|600。
            //   fontSize: 18, // 字体大小
            //   lineHeight: "130", // 行高
            //   textBorderColor: "red", // 文字本身的描边颜色。
            //   textBorderWidth: 5, // 文字本身的描边宽度。
            //   textShadowColor: "transparent", // 文字本身的阴影颜色。
            //   textShadowBlur: 0, // 文字本身的阴影长度。
            //   textShadowOffsetX: 0, // 文字本身的阴影 X 偏移。
            //   textShadowOffsetY: 0, //  文字本身的阴影 Y 偏移。
            // },
        },
        tooltip: {},

        legend: {},
        xAxis: {
          data: ["衬衫", "羊毛衫", "雪纺衫", "裤子", "高跟鞋", "袜子"],
        },
        yAxis: {},
        series: [
          {
            name: "销量",
            type: "bar",
            data: [5, 20, 36, 10, 10, 20],
          },
        ],
      });
    },
};
</script>
<style>
#main {
  width: 500px;
  height: 500px;
}
</style>
```

7. 配置项--tooltip

提示框组件，用于配置鼠标滑过或点击图表时的显示框

```
<template>
  <div class="about">
    <h1>This is an about page</h1>
    <div id="main"></div>
  </div>
</template>
<script>
import * as echarts from "echarts";
export default {
  mounted() {
```

```
    var myChart = echarts.init(document.getElementById("main"));
    myChart.setOption({
      title: {
          text: '主标题'
      },
      tooltip: {//提示框组件，用于配置鼠标滑过或点击图表时的显示框。
        show: true, // 是否显示
        trigger: 'axis', // 触发类型  'item'图形触发：散点图，饼图等无类目轴的图表中使用； 'axis'坐标轴触发；'none'：什么都不触发。
        axisPointer: { // 坐标轴指示器配置项。
            type: 'cross', // 'line' 直线指示器  'shadow' 阴影指示器  'none' 无指示器  'cross' 十字准星指示器。
          },
        // showContent: true, //是否显示提示框浮层，默认显示。
        // triggerOn: 'mouseover', // 触发时机'click'鼠标点击时触发。
        backgroundColor: 'rgba(50,50,50,0.7)', // 提示框浮层的背景颜色。
        borderColor: '#333', // 提示框浮层的边框颜色。
        borderWidth: 0, // 提示框浮层的边框宽。
        padding: 5, // 提示框浮层内边距，
        textStyle: { // 提示框浮层的文本样式。
            color: '#fff',
            fontStyle: 'normal',
            fontWeight: 'normal',
            fontFamily: 'sans-serif',
            fontSize: 14,
        },
        // 提示框浮层内容格式器，支持字符串模板和回调函数两种形式。
        // 模板变量有 {a}, {b}，{c}，分别表示系列名，数据名，数据值等
        // formatter: '{a}--{b} 的成绩是 {c}'
        formatter: function(arg) {
            return arg[0].name + '的分数是:' + arg[0].data
        }
      },

      legend: {},
      xAxis: {
        data: ["衬衫", "羊毛衫", "雪纺衫", "裤子", "高跟鞋", "袜子"],
      },
      yAxis: {},
      series: [
        {
          name: "销量",
          type: "bar",
          data: [5, 20, 36, 10, 10, 20],
        },
      ],
    });
  },
};
</script>
<style>
#main {
  width: 500px;
```

```
    height: 500px;
}
</style>
```

8.配置项--legend

图例组件展现了不同系列的标记，颜色和名字。可以通过点击图例控制哪些系列不显示。

```
<template>
  <div class="about">
    <h1>This is an about page</h1>
    <div id="main"></div>
  </div>
</template>
<script>
import * as echarts from "echarts";
export default {
  mounted() {
    var myChart = echarts.init(document.getElementById("main"));
    myChart.setOption({
      title: {
          text: '主标题'
      },
      tooltip: {

      },

      legend: {
        show: true, //是否显示
        icon: "circle",//图例样式
        // top: "55%", // 组件离容器的距离
        // bottom:"20%", // 组件离容器的距离
        // left 的值可以是像 20 这样的具体像素值，可以是像 '20%' 这样相对于容器高宽的百分比，也可以是 'left', 'center', 'right'
        // right: "5%",
        // left:"10%" // // 组件离容器的距离
        // padding: 5, // 图例内边距
        // itemWidth: 6, // 图例标记的图形宽度。
        // itemGap: 20, // 图例每项之间的间隔。
        // itemHeight: 14, // 图例标记的图形高度。
        // selectedMode: false, // 图例选择的模式，控制是否可以通过点击图例改变系列的显示状态。默认开启图例选择，可以设成 false 关闭。
        inactiveColor: "#fffddd", // 图例关闭时的颜色。
        textStyle: {//图例的公用文本样式。
          // color: "#aabbcc", // 文字的颜色。
          // fontStyle: "normal", // 文字字体的风格。'italic'
          // fontWeight: "normal", // 文字字体的粗细。 'normal' 'bold' 'bolder' 'lighter' 100 | 200 | 300 | 400...
          // fontFamily: "sans-serif", // 文字的字体系列。
          // fontSize: 12, // 文字的字体大小。
          // lineHeight: 20, // 行高。
```

```
            // backgroundColor: "transparent", // 文字块背景色。
            // borderColor: "transparent", // 文字块边框颜色。
            // borderWidth: 0, // 文字块边框宽度。
            // borderRadius: 0, // 文字块的圆角。
            // padding: 0, // 文字块的内边距
            // shadowColor: "transparent", // 文字块的背景阴影颜色
            // shadowBlur: 0, // 文字块的背景阴影长度。
            // shadowOffsetX: 0, // 文字块的背景阴影 X 偏移。
            // shadowOffsetY: 0, // 文字块的背景阴影 Y 偏移。
            // // width: 50, // 文字块的宽度。 默认
            // // height: 40, // 文字块的高度 默认
            // textBorderColor: "transparent", // 文字本身的描边颜色。
            // textBorderWidth: 0, // 文字本身的描边宽度。
            // textShadowColor: "transparent", // 文字本身的阴影颜色。
            // textShadowBlur: 0, // 文字本身的阴影长度。
            // textShadowOffsetX: 0, // 文字本身的阴影 X 偏移。
            // textShadowOffsetY: 0, // 文字本身的阴影 Y 偏移。
          }
        },
        xAxis: {
          data: ["衬衫", "羊毛衫", "雪纺衫", "裤子", "高跟鞋", "袜子"],
        },
        yAxis: {},
        series: [
          {
            name: "销量",
            type: "bar",
            data: [5, 20, 36, 10, 10, 20],
          },
        ],
      });
    },
  };
  </script>
  <style>
  #main {
    width: 500px;
    height: 500px;
  }
  </style>
```

9.柱状图基本设置

柱状图：一种图表类型,因为构成是由一根一根类似柱子的数据条组合而成的坐标平面,所以命名为柱状图。主要是用来反应对比数据之间的关系,也可以用来反应数据的变化趋势等等。

```
<template>
  <div class="about">
    <h1>This is an about page</h1>
    <!-- 2.echarts根结点根容器如果我们没有去指定当前容器的大小 echarts会按照0来进行处理 -->
    <div id="myecharts" ref="demoh"></div>
```

```
  </div>
</template>
<script>
import * as echarts from "echarts";
export default {
  mounted() {
    let myChart = echarts.init(this.$refs.demoh);
    let xData = ["美食", "数码", "日化", "蔬菜", "熟食"]; //x轴数据
    let yData = [88, 75, 20, 210, 35]; //y轴数据
    let option = {
      xAxis: {
        //配置x轴坐标参数
        data: xData,
        type: "category", //坐标轴类型。'value' 数值轴，适用于连续数据。
        // 'category' 类目轴，适用于离散的类目数据。为该类型时类目数据可自动从 series.data 或 dataset.source 中取，或者可通过 xAxis.data 设置类目数据。
        // 'time' 时间轴，适用于连续的时序数据，与数值轴相比时间轴带有时间的格式化，在刻度计算上也有所不同，例如会根据跨度的范围来决定使用月，星期，日还是小时范围的刻度。
        // 'log' 对数轴。适用于对数数据。
      },
      yAxis: {
        //配置y轴坐标参数
        type: "value", //同x轴的参数
      },
      series: [
        //系列 配置图表的类型
        {
          type: "bar",
          name: "销量", //系列名称，用于提示框组件的显示，
          data: yData,
        },
      ],
    };
    // 绘制图表 setOption 配置图表的配置项
    myChart.setOption(option);
  },
};
</script>
<style scoped>
#myecharts {
  width: 600px;
  height: 600px;
  border: 2px solid red;
}
</style>
```

10.柱状图效果实现

当基本的柱状图设置完之后我们来看一下 柱状图的更多设置 柱状图标记效果

最大值最小值平均值 通过markPoint进行设置

```vue
<template>
  <div class="about">
    <h1>This is an about page</h1>
    <div id="myecharts" ref="demoh"></div>
  </div>
</template>
<script>
import * as echarts from "echarts";
export default {
  mounted() {
    let myChart = echarts.init(this.$refs.demoh);
    let xData = ["美食", "数码", "日化", "蔬菜", "熟食"]; //x轴数据
    let yData = [88, 75, 20, 210, 35]; //y轴数据
    let option = {
      xAxis: {
        //配置x轴坐标参数
        data: xData,
        type: "category", //坐标轴类型。'value' 数值轴，适用于连续数据。
        // 'category' 类目轴，适用于离散的类目数据。为该类型时类目数据可自动从 series.data 或 dataset.source 中取，或者可通过 xAxis.data 设置类目数据。
        // 'time' 时间轴，适用于连续的时序数据，与数值轴相比时间轴带有时间的格式化，在刻度计算上也有所不同，例如会根据跨度的范围来决定使用月，星期，日还是小时范围的刻度。
        // 'log' 对数轴。适用于对数数据。
      },
      yAxis: {
        //配置y轴坐标参数
        type: "value", //同x轴的参数
      },
      series: [
        //系列 配置图表的类型
        {
          type: "bar",
          name: "销量", //系列名称，用于提示框组件的显示，
          data: yData,
          ///////////最大值最小值/////////////////
          markPoint: {
            //图表标注。
            data: [
              //标注的数据数组。每个数组项是一个对象
              {
                type: "max", //直接用 type 属性标注系列中的最大值，最小值。
                name: "最大值",
              },
              {
                type: "min",
                name: "最小值",
              },
            ],
          },
          ///////////最大值最小值/////////////////
          ///////////平均值/////////////////
          markLine: {
            //图表标线
```

```
            data: [
              //标线的数据数组。
              {
                type: "average",
                name: "平均值",
              },
            ],
          },
          ///////////平均值/////////////////
        },
      ],
    };
    // 绘制图表 setOption 配置图表的配置项
    myChart.setOption(option);
  },
};
</script>
<style scoped>
#myecharts {
  width: 600px;
  height: 600px;
  border: 2px solid red;
}
</style>
```

11.柱状图效果实现2--xAxis，yAxis

水平柱状图

通过设置xAxis yAxis中的type属性值来进行设置

barWidth : xx,设置柱图宽度

设置单独柱子的颜色

```
<template>
  <div class="about">
    <h1>This is an about page</h1>
    <div id="myecharts" ref="demoh"></div>
  </div>
</template>
<script>
import * as echarts from "echarts";
export default {
  mounted() {
    let myChart = echarts.init(this.$refs.demoh);
    let xData = ["美食", "数码", "日化", "蔬菜", "熟食"]; //x轴数据
    let yData = [88, 75, 20, 210, 35]; //y轴数据
    let option = {
      xAxis: {
        type: "value", //数值轴
        //坐标轴类型。'value' 数值轴，适用于连续数据。
```

```
        // 'category' 类目轴，适用于离散的类目数据。为该类型时类目数据可自动从 series.data 或 dataset.source 中取，或者可通过 xAxis.data 设置类目数据。
        // 'time' 时间轴，适用于连续的时序数据，与数值轴相比时间轴带有时间的格式化，在刻度计算上也有所不同，例如会根据跨度的范围来决定使用月，星期，日还是小时范围的刻度。
        // 'log' 对数轴。适用于对数数据。
      },
      yAxis: {
        data: xData,
        type: "category", //设置y为类目轴
      },
      series: [
        {
          type: "bar",
          name: "销量",
          data: yData,
          barWidth: 50, //设置宽度
          // color:"red",//设置颜色
          // 单独设置每个柱子的颜色
          itemStyle: {
            normal: {
              //每根柱子颜色设置
              color: function (params) {
                let colorList = [
                  "#c23531",
                  "#2f4554",
                  "#61a0a8",
                  "#d48265",
                  "#91c7ae",
                ];
                return colorList[params.dataIndex];
              },
            },
          },
          markPoint: {
            data: [
              {
                type: "max",
                name: "最大值",
              },
              {
                type: "min",
                name: "最小值",
              },
            ],
          },

          markLine: {
            data: [
              {
                type: "average",
                name: "平均值",
              },
            ],
```

```
            },
          },
        ],
      };
      // 绘制图表 setOption 配置图表的配置项
      myChart.setOption(option);
    },
  };
  </script>
  <style scoped>
  #myecharts {
    width: 600px;
    height: 600px;
    border: 2px solid red;
  }
  </style>
```

12.饼状图基本设置

饼状图是用整个圆表示总体的数量或整体值“1”，用圆内各个扇形的大小表示各部分数量或该部分占总体的百分比。一般由标题（包括单位）、图例和数据等组成。 1.主要运用在对数据进行比较分析的时候，既可以表示绝对量，又可以表示相对量。 2.比柱形图等好在：数据更为清晰，各部分占总体的比重大小更为直观，可谓一目了然

```
<template>
  <div ref="myChart" id="myChart"></div>
</template>

<script>
import * as echarts from "echarts"
export default {
  mounted(){
    // 1.初始化
    let myChart=echarts.init(this.$refs.myChart)
    // 2.设置echarts数据
   let data=[
        { value: 67, name: '美食' },
        { value: 85, name: '日化' },
        { value: 45, name: '数码' },
        { value: 98, name: '家电' }
      ]

    // 3.设置配置项
    let option={
        title: {
    text: '饼状图',
    subtext: '基本设置',
    left: 'center'//设置位置居中
  },
  tooltip: {
    trigger: 'item'//触发类型item数据项图形触发
  },
```

```
    legend: {
      orient: 'vertical',//图例列表的布局朝向vertical纵向
      left: 'left'
    },
    series: [
      {
        name: '销售量',
        type: 'pie',//饼图主要用于表现不同类目的数据在总和中的占比。每个的弧度表示数据数量的比例。
        data
      }
    ]
      }
      // 4.设置图表绘制图表
      myChart.setOption(option)
    }
  }
  </script>

  <style>
  #myChart{
    width: 500px;
    height: 500px;
    border: 1px solid red;
  }
  </style>
```

13.饼状图效果实现

但是饼状图还有更多的效果

环形图 样式等内容设置

```
  <template>
    <div ref="myChart" id="myChart"></div>
  </template>

  <script>
  import * as echarts from "echarts"
  export default {
    mounted(){
      // 1.初始化
      let myChart=echarts.init(this.$refs.myChart)
      // 2.设置echarts数据
    //  let data=[
    //        { value: 67, name: '美食' },
    //        { value: 85, name: '日化' },
    //        { value: 45, name: '数码' },
    //        { value: 98, name: '家电' }
    //     ]
     let data=[
          {
```

```
            value: 67,
            name: '美食',
            itemStyle:{
                        normal:{
                           color:'rgb(1,175,80)'
                             }
                           }
                         },
          {
            value: 85,
            name: '日化',
            itemStyle:{
                        normal:{
                           color:'rgb(255,175,80)'
                             }
                           }

          },
          {
            value: 45,
            name: '数码',
            itemStyle:{
                        normal:{
                           color:'rgb(1,0,80)'
                             }
                           }
            },
          {
            value: 98,
            name: '家电',
            itemStyle:{
                        normal:{
                           color:'rgb(30,50,70)'
                             }
                           }
            }
        ]
      // 单独设置每个颜色

      // 3.设置配置项
      let option={
          title: {
      text: '饼状图',
      subtext: '基本设置',
      left: 'center'//设置位置居中
    },
    tooltip: {
      trigger: 'item'//触发类型item数据项图形触发
    },
    legend: {
      orient: 'vertical',//图例列表的布局朝向vertical纵向
      left: 'left'
    },
```

```
    series: [
      {
        name: '销售量',
        type: 'pie',
        // 设置环形图
        radius: ['40%', '70%'],//饼图的半径。数组的第一项是内半径，第二项是外半径。
        // 设置环形图
        label: {//饼图图形上的文本标签
          show: true,
          position:"inside",//outside饼图扇区外侧inside饼图扇区内部center在饼图中心位置
          color:"yellow"
        },
        labelLine: {//标签的视觉引导线配置
          show: false
        },
        roseType: 'area',//是否展示成南丁格尔图，通过半径区分数据大小
         itemStyle: {//设置内容样式
          color: '#c23531',
          shadowBlur: 200,
          shadowColor: 'rgba(0, 0, 0, 0.5)'
        },
        data
      }
    ]
      }
      // 4.设置图表绘制图表
      myChart.setOption(option)
    }
  }
  </script>

  <style>
  #myChart{
    width: 500px;
    height: 500px;
    border: 1px solid red;
  }
  </style>
```

14.折线图基本设置

折线图是用折线将各个数据点标志连接起来的图表，用于展现数据的变化趋势。

不仅可以表示数量的多少，而且可以反映同一事物在不同时间里的发展变化的情况。 易于显示数据变化趋势，可以直观地反映这种变化以及各组之间的差别。

```
<template>
  <div ref="myChart" id="myChart"></div>
</template>
```

```vue

<script>
import * as echarts from "echarts"
export default {
  mounted(){
    // 1.初始化
    let myChart=echarts.init(this.$refs.myChart)
    // 2.设置数据
    let xData=['Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun']
    let data=[150, 230, 224, 218, 135, 147, 260]
    // 3.设置配置项
    let option = {
  xAxis: {
    type: 'category',
    data: xData
  },
  yAxis: {
    type: 'value'
  },
  series: [
    {
      data,
      type: 'line'//设置系列为折线图
    }
  ]
};
    // 4.设置图表绘制图表
    myChart.setOption(option)
  }
}
</script>

<style>
#myChart{
  width: 500px;
  height: 500px;
  border: 1px solid red;
}
</style>
```

15.折线图效果实现

设置平滑过渡样式 并且可以对内容进行颜色的填充 加上对应的标记点

```vue
<template>
  <div ref="myChart" id="myChart"></div>
</template>

<script>
import * as echarts from "echarts"
export default {
```

```
  mounted(){
    // 1.初始化
    let myChart=echarts.init(this.$refs.myChart)
    // 2.设置数据
    let xData=['Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun']
    let data=[150, 230, 224, 218, 135, 147, 260]
    // 3.设置配置项
    let option = {
  xAxis: {
    type: 'category',
    data: xData
  },
  yAxis: {
    type: 'value'
  },
  series: [
    {
      data,
      type: 'line',//设置系列为折线图
      smooth: true,//是否平滑曲线显示如果是 number 类型（取值范围 0 到 1），表示平滑程度，越小表示越接近折线段，反之则反。设为 true 时相当于设为 0.5
       areaStyle: {},//区域填充样式。设置后显示成区域面积图。
      markPoint: {//图表标注。
        data: [
          { type: 'max', name: 'Max' },
          { type: 'min', name: 'Min' }
        ]
      },
      markLine: {//图表标线。
        data: [{ type: 'average', name: 'Avg' }]
      }
    }
  ]
};
    // 4.设置图表绘制图表
    myChart.setOption(option)
  }
}
</script>

<style>
#myChart{
  width: 500px;
  height: 500px;
  border: 1px solid red;
}
</style>
```

16.折线图堆叠效果

设置多折折线效果

```vue
<template>
  <div ref="myChart" id="myChart"></div>
</template>

<script>
import * as echarts from "echarts";
export default {
  mounted() {
    // 1.初始化
    let myChart = echarts.init(this.$refs.myChart);
    // 2.设置数据
    let xData = ["Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"];
    // 3.设置配置项
    let option = {
      xAxis: {
        type: "category",
        data: xData,
      },
      yAxis: {
        type: "value",
      },

      series: [
        {
          name: "美食",
          type: "line",
          stack: "num", //数据堆叠，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。
          data: [120, 132, 101, 134, 90, 230, 210],
          areaStyle: {}, //区域填充样式。设置后显示成区域面积图。
          emphasis: {
            //折线图的高亮状态。
            focus: "series", //聚焦当前高亮的数据所在的系列的所有图形。
          },
        },
        {
          name: "日化",
          type: "line",
          stack: "num",
          data: [220, 182, 191, 234, 290, 330, 310],
          areaStyle: {}, //区域填充样式。设置后显示成区域面积图。
          emphasis: {
            //折线图的高亮状态。
            focus: "series", //聚焦当前高亮的数据所在的系列的所有图形。
          },
        },
        {
          name: "熟食",
          type: "line",
          stack: "num",
          data: [150, 232, 201, 154, 190, 330, 410],
          areaStyle: {}, //区域填充样式。设置后显示成区域面积图。
          emphasis: {
```

```
            //折线图的高亮状态。
            focus: "series", //聚焦当前高亮的数据所在的系列的所有图形。
          },
        },
      ],
    };
    // 4.设置图表绘制图表
    myChart.setOption(option);
  },
};
</script>

<style>
#myChart {
  width: 500px;
  height: 500px;
  border: 1px solid red;
}
</style>
```

17.散点图基本效果设置

当存在大量数据点时，*散点图* 的 *作用* 尤为明显。 *散点图* 与折线图相似，而不同之处在于折线图通过将点或数据点相连来显示每一个变化。

```
<template>
  <div ref="myChart" id="myChart"></div>
</template>

<script>
import * as echarts from "echarts";
export default {
  mounted() {
    // 1.初始化
    let myChart = echarts.init(this.$refs.myChart);
    // 2.设置配置项
   let option = {
      xAxis: {},
      yAxis: {},
      series: [
        {
          symbolSize: 20,
          data: [
            [9.0, 7.04],
            [18.07, 4.33],
            [3.0, 9.65],
            [9.05, 8.23],
            [18.0, 9.76],
            [15.0, 7.56],
            [23.4, 5.31],
            [10.1, 7.47],
            [16.0, 8.26],
            [12.7, 3.53],
```

```
            [9.35, 7.2],
            [7.4, 8.2],
            [3.07, 4.82],
            [18.2, 6.83],
            [2.02, 4.47],
            [1.05, 3.33],
            [4.05, 4.96],
            [6.03, 7.24],
            [17.0, 6.55],
            [12.0, 8.84],
            [8.18, 5.82],
            [6.32, 5.68],
          ],
          type: "scatter",//散点图

        },
      ],
    };
    // 3.设置图表绘制图表
    myChart.setOption(option);
  },
};
</script>

<style>
#myChart {
  width: 500px;
  height: 500px;
  border: 1px solid red;
}
</style>
```

18.散点图效果实现

样式相关设置

```
<template>
  <div ref="myChart" id="myChart"></div>
</template>

<script>
import * as echarts from "echarts";
export default {
  mounted() {
    // 1.初始化
    let myChart = echarts.init(this.$refs.myChart);
    // 2.设置配置项
    let option = {
      xAxis: {},
```

```
        yAxis: {

        },
          tooltip: {},//提示框组件
        series: [
          {
            symbolSize: 20,
            data: [
              [9.0, 7.04],
              [18.07, 4.33],
              [3.0, 9.65],
              [9.05, 8.23],
              [18.0, 9.76],
              [15.0, 7.56],
              [23.4, 5.31],
              [10.1, 7.47],
              [16.0, 8.26],
              [12.7, 3.53],
              [9.35, 7.2],
              [7.4, 8.2],
              [3.07, 4.82],
              [18.2, 6.83],
              [2.02, 4.47],
              [1.05, 3.33],
              [4.05, 4.96],
              [6.03, 7.24],
              [17.0, 6.55],
              [12.0, 8.84],
              [8.18, 5.82],
              [6.32, 5.68],
            ],
            type: "scatter", //散点图
            // 圆形样式
            color: {//线性渐变，前四个参数分别是 x0, y0, x2, y2, 范围从 0 - 1，相当于在图形包围盒中的百分比
              type: "linear",
              x: 0,
              y: 0,
              x2: 1,
              y2: 0,
              colorStops: [
                {
                  offset: 0,
                  color: "#00CCFF", // 0% 处的颜色
                },
                {
                  offset: 1,
                  color: "rgba(255, 173, 119, 1)", // 100% 处的颜色
                },
              ],
              globalCoord: true, // 如果 globalCoord 为 `true`，则该四个值是绝对的像素位置
            },
            emphasis: {//高亮的图形和标签样式
```

```
            itemStyle: {
              borderColor: "rgba(102,205,46,0.30)",
              borderWidth: 30,
            },
          },

        },
      ],
    };
    // 3.设置图表绘制图表
    myChart.setOption(option);
  },
};
</script>

<style>
#myChart {
  width: 500px;
  height: 500px;
  border: 1px solid red;
}
</style>
```

19.配置项--grid

grid 为直角坐标系内绘图网格。可以在网格上绘制折线图，柱状图 散点图（气泡图）也就是设置图标离容器的距离样式

```
<template>
  <div class="about">
    <!-- 2.echarts根结点根容器如果我们没有去指定当前容器的大小 echarts会按照0来进行处理 -->
    <div id="myecharts" ref="demoh"></div>
  </div>
</template>
<script>
import * as echarts from "echarts";
export default {
  mounted() {
    let myChart = echarts.init(this.$refs.demoh);
    let xData = ["美食", "数码", "日化", "蔬菜", "熟食"];
    let yData = [88, 75, 20, 210, 35];
    let option = {
      xAxis: {
        data: xData,
        type: "category",
      },
      yAxis: {
        type: "value",
      },
      // grid配置项：图标离容器的距离
      // show:是否显示直角坐标系网格----------值:true?false
```

```
        // left:图表离容器左侧的距离----------------值:number?百分比
        // top:图表离容器顶部的距离----------------值:number?百分比
        // right:图表离容器右侧的距离--------------值:number?百分比
        // bottom:图表离容器底部的距离------------值:number?百分比
        // backgroundColor:网格背景色--------------值:rgba或#000000
        // borderColor:网格的边框颜色--------------值:rgba或#000000
        // borderWidth:网格的边框线宽-------------值:number
        grid: {
          show: true,
          left: "5%",
          top: "5%",
          right: "5%",
          bottom: "5%",
          backgroundColor: "rgba(224, 17, 17, 1)",
          borderColor: "rgba(96, 67, 67, 1)",
        },
        series: [
          {
            type: "bar",
            name: "销量",
            data: yData,
          },
        ],
      };
      myChart.setOption(option);
    },
  };
  </script>
  <style scoped>
  #myecharts {
    width: 600px;
    height: 600px;
    border: 2px solid red;
  }
  </style>
```

20.K 线图

K线图可以查看k线历史走势，近期趋势，是上涨还是下跌，是调整还是震荡。分析k线的高低点和相对高低点。方便对于数据的走势进行查看

基本设置

```
<template>
  <div ref="myChart" id="myChart"></div>
</template>

<script>
import * as echarts from "echarts";
```

```
export default {
  mounted() {
    // 1.初始化
    let myChart = echarts.init(this.$refs.myChart);
    // 2.设置配置项
    let option = {
      xAxis: {
        data: ["蔬菜", "水果", "熟食", "便捷食品"],
      },
      yAxis: {},
      series: [
        {
          type: "candlestick",//k线图
          data: [
            [20, 34, 10, 38],
            [40, 35, 30, 50],
            [31, 38, 33, 44],
            [38, 15, 5, 42],
          ],
        },
      ],
    };
    // 3.设置图表绘制图表
    myChart.setOption(option);
  },
};
</script>

<style>
#myChart {
  width: 500px;
  height: 500px;
  border: 1px solid red;
}
</style>
```

21 k线图效果优化

```
<template>
  <div ref="myChart" id="myChart"></div>
</template>

<script>
import * as echarts from "echarts";
export default {
  data(){
    return {
      data:[
      [20, 34, 10, 38],
      [40, 35, 30, 50],
      [31, 38, 33, 44],
      [38, 15, 5, 42],
```

```
    ]
    }
  },
  computed:{
    newarr(){
    let linstdata= this.data.map((v)=>{

        return v[0]
      })
      return linstdata
    }
  },
  mounted() {
    // 1.初始化
    let myChart = echarts.init(this.$refs.myChart);
    // 2.设置配置项
    let option = {
      xAxis: {
        data: ["蔬菜", "水果", "熟食", "便捷食品"],
      },
      yAxis: {},
      tooltip: {
        //设置提示框
        trigger: "axis",
        axisPointer: {
          type: "cross",
        },
      },
      series: [
        {
          type: "candlestick", //k线图
          data: this.data,
          itemStyle: {
            color: "#ec0000", //上涨的颜色
            color0: "#00da3c", //下跌的颜色
            borderColor: "#8A0000", //上涨的边框色
            borderColor0: "#008F28", //下跌的边框色
          },
          markPoint: {
            data: [
              {
                name: "最大值",
                type: "max",
                valueDim: "highest", //valueDim 指定是在哪个维度上的最大值、最小值、平均值
              },
              {
                name: "最小值",
                type: "min",
                valueDim: "lowest",
              },
              {
                name: "平均值",
                type: "average",
```

```
                valueDim: "close",
              },
            ],
          },
        },
        {
          name: "MA20",
          type: "line",
          data: this.newarr,
          smooth: true,
          lineStyle: {
            opacity: 0.5,
          },
        },
      ],
    };
    // 3.设置图表绘制图表
    myChart.setOption(option);
  },
};
</script>

<style>
#myChart {
  width: 500px;
  height: 500px;
  border: 1px solid red;
}
</style>
```

22.雷达图

基本设置

```
<template>
  <div ref="myChart" id="myChart"></div>
</template>

<script>
import * as echarts from "echarts";
export default {
  mounted() {
    // 1.初始化
    let myChart = echarts.init(this.$refs.myChart);
    // 2.设置配置项
    let option = {
      title: {
        text: "雷达图",
      },
      radar: [//雷达图坐标系组件，只适用于雷达图
        {
```

```
            // shape: 'circle',//设置及雷达图效果
            indicator: [//雷达图的指示器，用来指定雷达图中的多个变量（维度）
              { name: "蔬菜", max: 6500 },
              { name: "水果", max: 16000 },
              { name: "熟食", max: 30000 },
              { name: "数码", max: 38000 },
              { name: "家电", max: 52000 },
              { name: "日化", max: 25000 },
            ],
          },
        ],

        series: [
          {
            type: "radar", //雷达图
            data: [
              {
                value: [4200, 3000, 20000, 35000, 50000, 18000],
                name: "销量",
              },
            ],
          },
        ],
      };
      // 3.设置图表绘制图表
      myChart.setOption(option);
    },
  };
  </script>

  <style>
  #myChart {
    width: 500px;
    height: 500px;
    border: 1px solid red;
  }
  </style>
```

23.雷达图效果优化

```
<template>
  <div ref="myChart" id="myChart"></div>
</template>

<script>
import * as echarts from "echarts";
export default {
  mounted() {
    // 1.初始化
    let myChart = echarts.init(this.$refs.myChart);
    // 2.设置配置项
    let option = {
```

```
        title: {
          text: "雷达图",
        },
        radar: [//雷达图坐标系组件，只适用于雷达图
          {
            // shape: 'circle',//设置及雷达图效果
            indicator: [//雷达图的指示器，用来指定雷达图中的多个变量（维度）
              { name: "蔬菜", max: 6500 },
              { name: "水果", max: 16000 },
              { name: "熟食", max: 30000 },
              { name: "数码", max: 38000 },
              { name: "家电", max: 52000 },
              { name: "日化", max: 25000 },
            ],
            radius: 120,//b半径
            startAngle: 90,//坐标系起始角度，也就是第一个指示器轴的角度(可以让内容旋转)
            splitNumber: 10,//指示器轴的分割段数(内部的分割数量)。
            shape: "circle",//雷达图绘制类型
            axisName: {//雷达图每个指示器名称的配置项
              formatter: "【{value}】",//使用字符串模板，模板变量为指示器名称 {value}
              color: "#428BD4",
            },
            splitArea: {//坐标轴在 grid 区域中的分隔区域，默认不显示。
              areaStyle: {//分隔区域的样式设置。
                color: ["#77EADF", "#26C3BE", "#64AFE9", "#428BD4"],
                shadowColor: "rgba(0, 0, 0, 0.2)",
                shadowBlur: 10,
              },
            },
          },
        ],
        series: [
          {
            type: "radar", //雷达图
            symbol: "rect",//标记的图形。
            symbolSize: 12,//标记大小
            lineStyle: {
              type: "dashed",
            },
            data: [
              {
                value: [4200, 3000, 20000, 35000, 50000, 18000],
                name: "销量",
                areaStyle: {
                  //设置填充
                  color: "rgba(255, 228, 52, 0.6)",
                },
              },
            ],
          },
        ],
      };
      // 3.设置图表绘制图表
```

```
    myChart.setOption(option);
  },
};
</script>

<style>
#myChart {
  width: 500px;
  height: 500px;
  border: 1px solid red;
}
</style>
```

24.漏斗图 基本设置

```
<template>
  <div ref="myChart" id="myChart"></div>
</template>

<script>
import * as echarts from "echarts";
export default {
  mounted() {
    // 1.初始化
    let myChart = echarts.init(this.$refs.myChart);
    // 2.设置配置项
    let option = {
      title: {
        text: "漏斗图",
      },
      tooltip: {//设置弹框
        trigger: "item",
        formatter: "{a} <br/>{b} : {c}%",
      },


      series: [
        {

          type: "funnel",//设置漏斗图
          data: [
            { value: 60, name: "美食" },
            { value: 40, name: "日化" },
            { value: 20, name: "数码" },
            { value: 80, name: "家电" },
            { value: 100, name: "蔬菜" },
          ],
        },
      ],
    };
    // 3.设置图表绘制图表
```

```
    myChart.setOption(option);
  },
};
</script>

<style>
#myChart {
  width: 500px;
  height: 500px;
  border: 1px solid red;
}
</style>
```

25 漏斗图效果实现

```
<template>
  <div ref="myChart" id="myChart"></div>
</template>

<script>
import * as echarts from "echarts";
export default {
  mounted() {
    // 1.初始化
    let myChart = echarts.init(this.$refs.myChart);
    // 2.设置配置项
    let option = {
      title: {
        text: "漏斗图",
      },
      tooltip: {//设置弹框
        trigger: "item",
        formatter: "{a} <br/>{b} : {c}%",
      },


      series: [
        {

          type: "funnel",//设置漏斗图
          left: "10%",//漏斗图组件离容器左侧的距离
          top: 60,//顶部距离
          bottom: 60,//底部距离
          // width: "80%",
          min: 0,//指定的数据最小值。
          max: 100,
          minSize: "0%",//数据最小值 min 映射的宽度。
          maxSize: "100%",
          sort: "ascending",//数据排序递减的    ascending递增 none根据数据
          gap: 2,//数据图形间距。
          label: {//提示信息位置
            show: true,
```

```
            position: "inside",
          },

          itemStyle: {//漏斗图样式
            borderColor: "red",
            borderWidth: 2,
          },
          emphasis: {//选中高亮的标签和图形样式。
            label: {
              fontSize: 30,
            },
          },
          data: [
               { value: 60, name: "美食" },
            { value: 40, name: "日化" },
            { value: 20, name: "数码" },
            { value: 80, name: "家电" },
            { value: 100, name: "蔬菜" },
          ],
        },
      ],
    };
    // 3.设置图表绘制图表
    myChart.setOption(option);
  },
};
</script>

<style>
#myChart {
  width: 500px;
  height: 500px;
  border: 1px solid red;
}
</style>
```

26.仪表盘

```
<template>
  <div ref="myChart" id="myChart"></div>
</template>

<script>
import * as echarts from "echarts"
export default {
  mounted(){
    let myEcharts=echarts.init(this.$refs.myChart)
    let options={
      series:[
```

```
        {
          type:"gauge",
          data:[
            {
              value:45,
              name:"提示信息"
            }
          ],
          detail:{
            valueAnimation:true
          },
          progress:{
            show:true
          }
        }
      ]
    }

    myEcharts.setOption(options)
  }

}
</script>

<style>
#myChart{
  width: 500px;
  height: 500px;
  border: 1px solid red;
}
</style>
```

27. 关系图

创建节点

```
<template>
  <div ref="myChart" id="myChart"></div>
</template>

<script>
import * as echarts from "echarts"
export default {
  data(){
    return {
      list:[//创建节点数据
        {
          name: "韦小宝",
          id: "1",
        },
        {
          name: "方怡",
```

```
            id: "2",
          },
          {
            name: "双儿",
            id: "3",
          },
          {
            name: "茅十八",
            id: "4",
          },
          {
            name: "吴六奇",
            id: "5",
          },
        ]
      }
    },
    mounted(){
      let myEcharts=echarts.init(this.$refs.myChart)

      let options = {
       series:[
          {
            type: 'graph',//图标类型为关系图用于展现节点以及节点之间的关系数据
             layout: 'force',//图的布局   引导布局
            data:this.list
          }
          ]
        }

      myEcharts.setOption(options)
    }

}
</script>

<style>
#myChart{
  width: 500px;
  height: 500px;
  border: 1px solid red;
}
</style>
```

增加节点样式

```
<template>
  <div ref="myChart" id="myChart"></div>
</template>

<script>
import * as echarts from "echarts"
export default {
```

```
  data(){
    return {
      list:[//创建节点数据
        {
          name: "韦小宝",
          id: "1",
          symbolSize: 30,//节点大小
          symbol:'circle',//节点形状，
        },
        {
          name: "方怡",
          id: "2",
          symbolSize: 30,//节点大小
          symbol:'circle',//节点形状，
        },
        {
          name: "双儿",
          id: "3",
          symbolSize: 30,//节点大小
          symbol:'circle',//节点形状，
        },
        {
          name: "茅十八",
          id: "4",
          symbolSize: 30,//节点大小
          symbol:'circle',//节点形状，
        },
        {
          name: "吴六奇",
          id: "5",
          symbolSize: 30,//节点大小
          symbol:'circle',//节点形状，
        },
      ]
    }
  },
  mounted(){
    let myEcharts=echarts.init(this.$refs.myChart)

    let options = {
     series:[
        {
          type: 'graph',//图标类型为关系图用于展现节点以及节点之间的关系数据
           layout: 'force',//图的布局  引导布局
          data:this.list,
            itemStyle: {//节点的样式
              color: "#95dcb2"
            },
            label: {//图形上的文本标签
              show: true,
              position: "bottom",//位置底部
              distance: 5,//距离图形元素的距离
              fontSize: 18,
```

```
                align: "center",//文字水平对齐方式
              },
          }
          ]
        }

      myEcharts.setOption(options)
    }

}
</script>

<style>
#myChart{
  width: 500px;
  height: 500px;
  border: 1px solid red;
}
</style>
```

创建关系数据与图

```
<template>
  <div ref="myChart" id="myChart"></div>
</template>

<script>
import * as echarts from "echarts";
export default {
  data() {
    return {
      list: [
        //创建节点数据
        {
          name: "韦小宝",
          id: "1",
          symbolSize: 30, //节点大小
          symbol: "circle", //节点形状，
        },
        {
          name: "方怡",
          id: "2",
          symbolSize: 30, //节点大小
          symbol: "circle", //节点形状，
        },
        {
          name: "双儿",
          id: "3",
          symbolSize: 30, //节点大小
          symbol: "circle", //节点形状，
        },
        {
          name: "茅十八",
```

```
            id: "4",
            symbolSize: 30, //节点大小
            symbol: "circle", //节点形状，
          },
          {
            name: "吴六奇",
            id: "5",
            symbolSize: 30, //节点大小
            symbol: "circle", //节点形状，
          },
        ],
        num: [
          //关系数据
          {
            source: "1", //边的源节点名称的字符串
            target: "2", //边的目标节点名称的字符串
            relation: {
              name: "老婆",
              id: "1",
            },
          },
          {
            source: "1",
            target: "3",
            relation: {
              name: "老婆",
              id: "1",
            },
          },
          {
            source: "1",
            target: "4",
            relation: {
              name: "兄弟",
              id: "1",
            },
          },
          {
            source: "4",
            target: "1",
            relation: {
              name: "兄弟",
              id: "1",
            },
          },
          {
            source: "3",
            target: "5",
            relation: {
              name: "义妹",
              id: "1",
            },
          },
```

```
        ],
      };
    },
    mounted() {
      let myEcharts = echarts.init(this.$refs.myChart);

      let options = {
        series: [
          {
            type: "graph", //图标类型为关系图用于展现节点以及节点之间的关系数据
            layout: "force", //图的布局  引导布局
            data: this.list,
            itemStyle: {
              //节点的样式
              color: "#95dcb2",
            },
            label: {
              //图形上的文本标签
              show: true,
              position: "bottom", //位置底部
              distance: 5, //距离图形元素的距离
              fontSize: 18,
              align: "center", //文字水平对齐方式
            },
            force: {
              //设置间距
              repulsion: 100, //点之间的距离
              gravity: 0.01, //设置距离中心点位置
              edgeLength: 200, //边的两个节点之间的距离
            },
            links: this.num, //节点间的关系数据

            edgeLabel: {
              //标签
              show: true,
              position: "middle", //标签位置线的中点
              fontSize: 12,
              formatter: (params) => {
                //标签内容格式设置内容
                return params.data.relation.name;
              },
            },
            edgeSymbol: ["circle", "arrow"], //边两边的类型

            autoCurveness: 0.01, //针对节点之间存在多边的情况，自动计算各边曲率
          },
        ],
      };

      myEcharts.setOption(options);
    },
  };
  </script>
```

```
138
139   <style>
140   #myChart {
141     width: 500px;
142     height: 500px;
143     border: 1px solid red;
144   }
145   </style>
```

28.数据区域缩放

用于区域缩放，从而能自由关注细节的数据信息，或者概览数据整体，或者去除离群点的影响。

```
1   <template>
2     <div ref="myChart" id="myChart"></div>
3   </template>
4
5   <script>
6   import * as echarts from "echarts"
7   export default {
8     mounted(){
9       let myChart=echarts.init(this.$refs.myChart)
10      let xData=['Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun']
11      let data=[150, 230, 224, 218, 135, 147, 260]
12      let option = {
13    xAxis: {
14      type: 'category',
15      data: xData
16    },
17    yAxis: {
18      type: 'value'
19    },
20    series: [
21      {
22        data,
23        type: 'line',
24        smooth: true,
25        markPoint: {
26          data: [
27            { type: 'max', name: 'Max' },
28            { type: 'min', name: 'Min' }
29          ]
30        },
31        markLine: {
32          data: [{ type: 'average', name: 'Avg' }]
33        }
34      },
35
36    ],
37      dataZoom: [//用于区域缩放，从而能自由关注细节的数据信息，或者概览数据整体，或者去除离群点的影响。
38      {
39        type: 'slider',//滑动条型数据区域缩放组件
40        xAxisIndex: 0,//x轴设置
```

```
        filterMode: 'none'//设置-----数据过滤不过滤数据，只改变数轴范围
      },
      {
        type: 'slider',
        yAxisIndex: 0,//y轴设置
        filterMode: 'none'
      },

    ],
  };
      // 4.设置图表绘制图表
      myChart.setOption(option)
    }
  }
  </script>

  <style>
  #myChart{
    width: 500px;
    height: 500px;
    border: 1px solid red;
  }
  </style>
```

29.基本树形图

树图主要用来可视化树形数据结构，是一种特殊的层次类型，具有唯一的根节点，左子树，和右子树。

```
  <template>
    <div ref="myChart" id="myChart"></div>
  </template>

  <script>
  import * as echarts from "echarts";
  export default {
    mounted() {
      let myEcharts = echarts.init(this.$refs.myChart);
      let data = {// 注意，最外层是一个对象，代表树的根节点
        name: "层级1",// 节点的名称，当前节点 label 对应的文本
        children: [
          {
            name: "层级2",
            children: [
              {
                name: "层级3-1",
                children: [//子节点
                  { name: "数据1", value: 3938 }, // value 值，只在 tooltip 中显示
                  { name: "数据2", value: 3812 },
                  { name: "数据3", value: 6714 },
                  { name: "数据4", value: 743 },
                ],
              },
              {
```

```
                name: "层级3-2",
                children: [
                  { name: "数据1", value: 3534 },
                  { name: "数据2", value: 5731 },
                  { name: "数据3", value: 7840 },
                  { name: "数据4", value: 5914 },
                  { name: "数据5", value: 3416 },
                ],
              },
            ],
          },
        ],
      };

      let options = {
        tooltip: {//提示框
          trigger: "item",//触发时机
        },
        series: [
          {
            type: "tree",//树图
            data: [data],
            top: "1%",//tree组件离容器顶部的距离
            left: "7%",
            bottom: "1%",
            right: "20%",
            symbolSize: 10,//标记的大小
            label: {//描述了每个节点所对应的文本标签的样式。
              position: "left",//标签的位置。
              verticalAlign: "middle",//文字垂直对齐方式
              align: "right",//文字水平对齐方式
              fontSize: 9,
            },
            leaves: {//叶子节点的特殊配置
              label: {//了叶子节点所对应的文本标签的样式
                position: "right",
                verticalAlign: "middle",
                align: "left",
              },
            },
            emphasis: {//树图中个图形和标签高亮的样式。
              focus: "descendant",//聚焦所有子孙节点
            },
            expandAndCollapse: true,//子树折叠和展开的交互由于绘图区域是有限的，而通常一个树图的节点可能会比较多，
            // 这样就会出现节点之间相互遮盖的问题。为了避免这一问题，可以将暂时无关的子树折叠收起，
            // 等到需要时再将其展开。如上面径向布局树图示例，节点中心用蓝色填充的就是折叠收起的子树，
            // 可以点击将其展开。
            animationDuration: 550,
            animationDurationUpdate: 750,
          },
        ],
      };
```

```
78
79        myEcharts.setOption(options);
80      },
81    };
82    </script>
83
84    <style>
85    #myChart {
86      width: 500px;
87      height: 500px;
88      border: 1px solid red;
89    }
90    </style>
```

30方向切换树形图

```
1
2
3   <template>
4     <div ref="myChart" id="myChart"></div>
5   </template>
6
7   <script>
8   import * as echarts from "echarts";
9   export default {
10    mounted() {
11      let myEcharts = echarts.init(this.$refs.myChart);
12      let data = {// 注意，最外层是一个对象，代表树的根节点
13        name: "层级1",// 节点的名称，当前节点 label 对应的文本
14        children: [
15          {
16            name: "层级2",
17            children: [
18              {
19                name: "层级3-1",
20                children: [//子节点
21                  { name: "数据1", value: 3938 }, // value 值，只在 tooltip 中显示
22                  { name: "数据2", value: 3812 },
23                  { name: "数据3", value: 6714 },
24                  { name: "数据4", value: 743 },
25                ],
26              },
27              {
28                name: "层级3-2",
29                children: [
30                  { name: "数据1", value: 3534 },
31                  { name: "数据2", value: 5731 },
32                  { name: "数据3", value: 7840 },
33                  { name: "数据4", value: 5914 },
34                  { name: "数据5", value: 3416 },
35                ],
36              },
```

```
            ],
          },
        ],
      };

      let options = {
        tooltip: {//提示框
          trigger: "item",//触发时机
        },
        series: [
          {
            type: "tree",
            data: [data],
            top: "1%",
            left: "7%",
            bottom: "1%",
            right: "20%",
            symbolSize: 10,
              //树图中 正交布局 的方向
            // 水平 方向的 从左到右'LR'，从右到左'RL'；
            // 以及垂直方向的 从上到下'TB'，从下到上'BT'
            orient: 'BT',
            label: {
              position: "bottom",
               rotate: 90,//文字旋转
              verticalAlign: "middle",
              align: "right",
              fontSize: 9,
            },
            leaves: {
              label: {
                position: "right",
                verticalAlign: "middle",
                align: "left",
              },
            },
            emphasis: {
              focus: "descendant",
            },
            expandAndCollapse: true,

            animationDuration: 550,
            animationDurationUpdate: 750,
          },
        ],
      };

      myEcharts.setOption(options);
    },
  };
  </script>

  <style>
```

```
#myChart {
  width: 500px;
  height: 500px;
  border: 1px solid red;
}
</style>
```

31.数据排序

```
<template>
  <div class="about">
    <h1>This is an about page</h1>
    <!-- 2.echarts根结点根容器如果我们没有去指定当前容器的大小 echarts会按照0来进行处理 -->
    <div id="myecharts" ref="demoh"></div>
  </div>
</template>
<script>
import * as echarts from "echarts";
export default {
  mounted() {
    let myChart = echarts.init(this.$refs.demoh);

    let option = {
      dataset: [
        //数据集组件用于单独的数据集声明，从而数据可以单独管理，被多个组件复用，并且可以自由指定数据到视觉的映射。这在不少场景下能带来使用上的方便。
        {
          dimensions: ["分类", "数量"], //设置分类数据
          source: [
            //原始数据。一般来说，原始数据表达的是二维表。
            ["Hannah Krause", 41],
            ["Zhao Qian", 20],
            ["Jasmin Krause ", 52],
            ["Li Lei", 37],
            ["Karle Neumann", 25],
            ["Adrian Groß", 19],
            ["Mia Neumann", 71],
          ],
        },
        {
          transform: {
            //数据改变
            type: "sort", //按照大小排序
            config: { dimension: "数量", order: "desc" }, //"sort" 数据转换器的“条件”
          },
        },
      ],
      xAxis: {
        type: "category",
```

```
          axisLabel: {
            //坐标轴刻度标签的相关设置。
            interval: 0, //坐标轴刻度标签的显示间隔，在类目轴中有效。
            rotate: 30, //刻度标签旋转的角度
          },
        },
        yAxis: {},
        series: [
          //系列 配置图表的类型
          {
            type: "bar",
            encode: {
              //可以定义 data 的哪个维度被编码成什么。
              x: "分类", //x映射内容
              y: "数量",
            },
            datasetIndex: 1,
          },
        ],
      };
      // 绘制图表 setOption 配置图表的配置项
      myChart.setOption(option);
    },
  };
  </script>
  <style scoped>
  #myecharts {
    width: 600px;
    height: 600px;
    border: 2px solid red;
  }
  </style>
```

32.内置主题

echarts中默认主题有两个:light、dark

echarts.init(选取容器dom,'主题')

```
<template>
  <div class="about">
    <h1>This is an about page</h1>

    <div id="myecharts" ref="demoh"></div>
  </div>
</template>
<script>
import * as echarts from "echarts";
export default {
  mounted() {
    //echarts中默认主题有两个:light、dark
```

```
      let myChart = echarts.init(this.$refs.demoh,"dark");
      let xData = ["美食", "数码", "日化", "蔬菜", "熟食"];
      let yData = [88, 75, 20, 210, 35];
      let option = {
        xAxis: {
          data: xData,
          type: "category",
        },
        yAxis: {
          type: "value",
        },
        series: [
          {
            type: "bar",
            name: "销量",
            data: yData,
          },
        ],
      };

      myChart.setOption(option);
    },
  };
  </script>
  <style scoped>
  #myecharts {
    width: 600px;
    height: 600px;
    border: 2px solid red;
  }
  </style>
```

33.自定义主题

1.在主题编辑器中编辑主题

主题编辑器地址： https://echarts.apache.org/zh/theme-builder.html

2.下载对应主题json格式

3.创建js文件把刚才下载的文件写入并且暴露

```
let roma=你的主题json

export default roma
```

4.引用主题文件

```
import roma from "../assets/roma"
```

5.在init方法中使用主题

34.中国地图展示效果

1.准备echarts基本结构

2.设置中国地图的矢量数据创建js文件 （ 在其中创建变量接受json数据 并且暴露 ）

地图数据下载地址： https://datav.aliyun.com/portal/school/atlas/area_selector

3.在组件中获取地图矢量数据（ 引用数据json ）

4.使用地图数据创建地图

```
<template>
  <div class="about">
    <h1>This is an about page</h1>

    <div id="myecharts" ref="demoh"></div>
  </div>
</template>
<script>
import * as echarts from "echarts";
// 引用的就是中国各省份的矢量数据
import cmap from "../assets/roma"
export default {
  mounted() {

    let myChart = echarts.init(this.$refs.demoh);
   echarts.registerMap("chinaMap",cmap)//使用 registerMap 注册的地图名称。
    let option = {
     geo:{//地理坐标系组件。地理坐标系组件用于地图的绘制
       type:"map",
       map:"chinaMap",//使用 registerMap 注册的地图名称
      //  默认设置完地图是固定死的不能拖动
      roam:true,//否开启鼠标缩放和平移漫游。默认不开启。
      zoom :10,//当前视角的缩放比例。越大比例越大
      center:[108.956239,34.268309],//当前视角的中心点，用经纬度表示108.956239,34.268309
      label:{//地图上显示文字提示信息
        show:true,
        color:"#ff6600",
        fontSize:10//字体大小
      },
      itemStyle:{//地图区域的多边形 图形样式。
        areaColor:"#ff6600"//地图区域的颜色。
      }
     }
    };

    myChart.setOption(option);
  },
};
</script>
```

```
<style scoped>
#myecharts {
  width: 600px;
  height: 600px;
  border: 2px solid red;
}
</style>
```

35.省份地图显示

同中国地图使用方式 就是切换地图数据即可

36.地图标记设置与效果

```
<template>
  <div class="about">
    <h1>This is an about page</h1>

    <div id="myecharts" ref="demoh"></div>
  </div>
</template>
<script>
import * as echarts from "echarts";

import cmap from "../assets/roma";
export default {
  mounted() {
    // 设置气泡点数据
    let data = [
      {
        value: [108.956239, 34.268309],
      },
    ];

    let myChart = echarts.init(this.$refs.demoh);
    echarts.registerMap("chinaMap", cmap);
    let option = {
      geo: {
        type: "map",
        map: "chinaMap",

        roam: true,
      },
      series: [
        {
          type: "effectScatter", //带有涟漪特效动画的散点（气泡）图。利用动画特效可以将某些想要突出的数据进行视觉突出。
          coordinateSystem: "geo", //使用什么坐标系geo使用地理坐标系
          data,
          // 这个时候地图上就会有点的涟漪效果
          rippleEffect: {
            //涟漪特效相关配置。
```

```
              number: 2, //波纹的数量。
              scale: 4, //动画中波纹的最大缩放比例
            },
            //  label:{
            //    show:true
            //  },
            itemStyle: {
              color: "red",
            },
          },
          // 也可以绘制点效果
          {
            symbolSize: 20,
            data: [
              {
                name: "北京市", // 数据项名称，在这里指地区名称
                value: [
                  // 数据项值
                  116.46, // 地理坐标，经度
                  39.92, // 地理坐标，纬度
                  340, // 北京地区的数值
                ],
              },
            ],
            type: "scatter",
            coordinateSystem: "geo",//// series坐标系类型
          },
        ],
      };

      myChart.setOption(option);
    },
  };
  </script>
  <style scoped>
  #myecharts {
    width: 600px;
    height: 600px;
    border: 2px solid red;
  }
  </style>
```

37.图表自适应大小

当浏览器大小改变的时候 我们需要让图表一同改变 这个时候就会用到图表自适应大小

```
<template>
  <div ref="myChart" id="myChart"></div>
</template>

<script>
import * as echarts from "echarts";
export default {
```

```
  mounted() {
    let myChart = echarts.init(this.$refs.myChart);
    let xData = ["Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"];
    let option = {
      xAxis: {
        type: "category",
        data: xData,
      },
      yAxis: {
        type: "value",
      },

      series: [
        {
          name: "美食",
          type: "line",
          stack: "num",
          data: [120, 132, 101, 134, 90, 230, 210],
          areaStyle: {},
          emphasis: {
            focus: "series",
          },
        },
        {
          name: "日化",
          type: "line",
          stack: "num",
          data: [220, 182, 191, 234, 290, 330, 310],
          areaStyle: {},
          emphasis: {
            focus: "series",
          },
        },
        {
          name: "熟食",
          type: "line",
          stack: "num",
          data: [150, 232, 201, 154, 190, 330, 410],
          areaStyle: {},
          emphasis: {
            focus: "series",
          },
        },
      ],
    };
    myChart.setOption(option);

    // 监听页面的大小
    window.addEventListener("resize", () => {
      myChart.resize()
    });
  },
};
```

```
</script>

<style>
#myChart {
  width: 100%;
  height: 500px;
  border: 1px solid red;
}
</style>
```

38.图表加载动画效果

myChart.showLoading();开始等待

myChart.hideLoading();关闭等待

1.设置json-server模拟数据

（1）全局下载 npm install -g json-server

（2）新建mock文件夹 并且在其中创建json文件 设置数据

（3）终端cd到mock文件夹下 启动 json-server --watch xx.json --port 端口号

2.页面请求数据并且设置等待效果

```
<template>
  <div ref="myChart" id="myChart"></div>
</template>

<script>
import * as echarts from "echarts";
import axios from "axios";
// import {mapData} from "../assets/mapData.js"
export default {
  data() {
    return {
      echartsData: {},
    };
  },
  methods: {
    // 获取json-server数据
    async linkData() {
      let mapnum = await axios({ url: "http://localhost:3000/one" });
      console.log(mapnum.data);
      this.echartsData = mapnum.data;
    },
  },

  mounted() {
    // 1.初始化
    let myChart = echarts.init(this.$refs.myChart);
    // 设置开始等待
    myChart.showLoading();

```

```
      // 调用数据请求方法
      this.linkData().then(() => {
        myChart.hideLoading();
        // 2.设置echarts数据

        let option = {
          title: {
            text: "饼状图",
            subtext: "基本设置",
            left: "center", //设置位置居中
          },
          tooltip: {
            trigger: "item", //触发类型item数据项图形触发
          },
          legend: {
            orient: "vertical", //图例列表的布局朝向vertical纵向
            left: "left",
          },
          series: [
            {
              name: "销售量",
              type: "pie",
              // 设置环形图
              radius: ["40%", "70%"], //饼图的半径。数组的第一项是内半径，第二项是外半径。
              // 设置环形图
              label: {
                //饼图图形上的文本标签
                show: true,
                position: "inside", //outside饼图扇区外侧inside饼图扇区内部center在饼图中心位置
                color: "yellow",
              },
              labelLine: {
                //标签的视觉引导线配置
                show: false,
              },
              roseType: "area", //是否展示成南丁格尔图，通过半径区分数据大小
              itemStyle: {
                //设置内容样式
                color: "#c23531",
                shadowBlur: 200,
                shadowColor: "rgba(0, 0, 0, 0.5)",
              },
              data: this.echartsData,
            },
          ],
        };
        // 4.设置图表绘制图表
        myChart.setOption(option);
      });
    },
  };
  </script>

```

```
<style>
#myChart {
  width: 500px;
  height: 500px;
  border: 1px solid red;
}
</style>
```

39.图表动画配置

```
<template>
  <div class="about">
    <h1>This is an about page</h1>
    <div id="myecharts" ref="demoh"></div>
  </div>
</template>
<script>
import * as echarts from "echarts";
export default {
  mounted() {
    let myChart = echarts.init(this.$refs.demoh);
    let xData = ["美食", "数码", "日化", "蔬菜", "熟食"];
    let yData = [88, 75, 20, 210, 35];
    let option = {
      animation:true,//是否开启动画。
      animationThreshold:5,//是否开启动画的阈值，当单个系列显示的图形数量大于这个阈值时会关闭动画。
      animationDuration:2000,//初始动画的时长
      animationEasing:"linear",//初始动画的缓动效果。官方更多解释：
https://echarts.apache.org/examples/zh/editor.html?c=line-easing
      animationDelay:1000,//初始动画的延迟
      xAxis: {
        type: "value",
      },
      yAxis: {
        data: xData,
        type: "category",
      },
      series: [
        {
          type: "bar",
          name: "销量",
          data: yData,
          barWidth: 50,
          itemStyle: {
            normal: {
              color: function (params) {
                let colorList = [
                  "#c23531",
                  "#2f4554",
                  "#61a0a8",
                  "#d48265",
                  "#91c7ae",
```

```
                ];
                return colorList[params.dataIndex];
              },
            },
          },
          markPoint: {
            data: [
              {
                type: "max",
                name: "最大值",
              },
              {
                type: "min",
                name: "最小值",
              },
            ],
          },

          markLine: {
            data: [
              {
                type: "average",
                name: "平均值",
              },
            ],
          },
        },
      ],
    };
    myChart.setOption(option);
  },
};
</script>
<style scoped>
#myecharts {
  width: 600px;
  height: 600px;
  border: 2px solid red;
}
</style>
```

40.echarts 事件

ECharts 中我们可以通过监听用户的操作行为来回调对应的函数。

ECharts 通过 **on** 方法来监听用户的行为，例如监控用户的点击行为。

```
<template>
  <div class="about">
    <h1>This is an about page</h1>
    <div id="myecharts" ref="demoh"></div>
```

```
    </div>
  </template>
  <script>
  import * as echarts from "echarts";
  export default {
    mounted() {
      let myChart = echarts.init(this.$refs.demoh);
      // 事件
      // ECharts 中我们可以通过监听用户的操作行为来回调对应的函数。

      // ECharts 通过 on 方法来监听用户的行为，例如监控用户的点击行为。
      myChart.on("click", function (params) {
        // 在用户点击后控制台打印数据的名称
        // params对象的属性
        // componentType 当前点击的图形元素所属的组件名称
        // seriesType 系列类型

        // seriesName 系列名称。
        // name数据名，类目名
        // data传入的原始数据项
        // value传入的数据值

        console.log(params);
      });

      let xData = ["美食", "数码", "日化", "蔬菜", "熟食"];
      let yData = [88, 75, 20, 210, 35];
      let option = {
        xAxis: {
          type: "value",
        },
        yAxis: {
          data: xData,
          type: "category",
        },
        series: [
          {
            type: "bar",
            name: "销量",
            data: yData,
            barWidth: 50,
            itemStyle: {
              normal: {
                color: function (params) {
                  let colorList = [
                    "#c23531",
                    "#2f4554",
                    "#61a0a8",
                    "#d48265",
                    "#91c7ae",
                  ];
                  return colorList[params.dataIndex];
                },
```

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

          markLine: {
            data: [
              {
                type: "average",
                name: "平均值",
              },
            ],
          },
        },

      ],
    };
    myChart.setOption(option);
  },
};
</script>
<style scoped>
#myecharts {
  width: 600px;
  height: 600px;
  border: 2px solid red;
}
</style>
```

有多个图形怎么监听呢？

使用 query 只对指定的组件的图形元素的触发回调：

```
chart.on(eventName, query, handler);
```

```
chart.on('click', 'series', function () {...});
chart.on('click', 'series.line', function () {...});
chart.on('click', 'dataZoom', function () {...});
chart.on('click', 'xAxis.category', function () {...});
```

下面就在添加一个折线图

```
<template>
```

```
    <div class="about">
      <h1>This is an about page</h1>
      <div id="myecharts" ref="demoh"></div>
    </div>
  </template>
  <script>
  import * as echarts from "echarts";
  export default {
    mounted() {
      let myChart = echarts.init(this.$refs.demoh);
      // 事件
      // ECharts 中我们可以通过监听用户的操作行为来回调对应的函数。

      // ECharts 通过 on 方法来监听用户的行为，例如监控用户的点击行为。
      // myChart.on("click", function (params) {
        // 在用户点击后控制台打印数据的名称
        // params对象的属性
        // componentType 当前点击的图形元素所属的组件名称
        // seriesType 系列类型

        // seriesName 系列名称。
        // name数据名，类目名
        // data传入的原始数据项
        // value传入的数据值

      //   console.log(params);
      // });


      // 只对折线图做出反应
      //   myChart.on("click", 'series.line',function (params) {
      //   console.log(params);
      // });
      // 只对某一项最做出反应
      // 比如对折线图的数码项点击做出反应
      // myChart.on("click",{name:"数码"},function (params) {
      //   console.log(params);
      // });
      // 但是发现折线图柱状图都可以
      // 只对折线图生效
      myChart.on("click",{seriesIndex: 1,name:"数码"},function (params) {
        console.log(params);
      });

      let xData = ["美食", "数码", "日化", "蔬菜", "熟食"];
      let yData = [88, 75, 20, 210, 35];
      let option = {
        xAxis: {
          type: "value",
        },
        yAxis: {
          data: xData,
          type: "category",
```

```
        },
        series: [
          {
            type: "bar",
            name: "销量",
            data: yData,
            barWidth: 50,
            itemStyle: {
              normal: {
                color: function (params) {
                  let colorList = [
                    "#c23531",
                    "#2f4554",
                    "#61a0a8",
                    "#d48265",
                    "#91c7ae",
                  ];
                  return colorList[params.dataIndex];
                },
              },
            },
            markPoint: {
              data: [
                {
                  type: "max",
                  name: "最大值",
                },
                {
                  type: "min",
                  name: "最小值",
                },
              ],
            },

            markLine: {
              data: [
                {
                  type: "average",
                  name: "平均值",
                },
              ],
            },
          },
    // 在添加一个折线图
          {
            data: [150, 230, 224, 218, 135],
            type: "line", //设置系列为折线图
            smooth: true, //是否平滑曲线显示如果是 number 类型（取值范围 0 到 1），表示平滑程度，越小表示越接近折线段，反之则反。设为 true 时相当于设为 0.5

            markPoint: {
              //图表标注。
              data: [
```

```
                { type: "max", name: "Max" },
                { type: "min", name: "Min" },
              ],
            },
            markLine: {
              //图表标线。
              data: [{ type: "average", name: "Avg" }],
            },
          },
        ],
      };
      myChart.setOption(option);
    },
  };
  </script>
  <style scoped>
  #myecharts {
    width: 600px;
    height: 600px;
    border: 2px solid red;
  }
  </style>
```