

课程介绍

讲师：千锋教育——奚皓源

Vue3.0与Echarts5.0

- 1.课程介绍与目标
- 2.数据可视化介绍
 - 什么是数据可视化?
 - 数据可视化的展现形式
 - 数据可视化有什么用?
- 3.Echarts--商业级数据图表介绍
 - 1.什么是Echarts
 - 2. Echarts特点
- 4.项目演示
 - 1.项目功能演示
 - 2.项目启动与依赖安装
- 5.HelloWord 初体验
 - 1.echarts获取
 - 前期准备
 - 安装
 - 2.HelloWord
- 6.配置项--title配置
- 7.配置项--tooltip
- 8.配置项--legend
- 9.柱状图基本设置
- 10.柱状图效果实现
- 11.柱状图效果实现2--xAxis, yAxis
- 12.饼状图基本设置
- 13.饼状图效果实现
- 14.折线图基本设置
- 15.折线图效果实现
- 16.折线图堆叠效果
- 17.散点图基本效果设置
- 18.散点图效果实现
- 19.配置项--grid
- 20.K 线图
- 21 k线图效果优化
- 22.雷达图
- 23.雷达图效果优化
- 24.漏斗图 基本设置
- 25 漏斗图效果实现
- 26.仪表盘
- 27.关系图
 - 创建节点
 - 增加节点样式
 - 创建关系数据与图
- 28.数据区域缩放
- 29.基本树形图
- 30方向切换树形图
- 31.数据排序
- 32.内置主题
- 33.自定义主题
- 34.中国地图展示效果
- 35.省份地图显示
- 36.地图标记设置与效果
- 37.图表自适应大小
- 38.图表加载动画效果

- 39.图表动画配置
- 40.echarts 事件
- 41.vue3.0项目创建
 - 1.电脑上安装node.js
 - 2.全局下载项目脚手架
 - 3.创建项目
 - 4运行项目
- 42.项目初始化?
- 43项目分辨率响应式分析与实施
 - 项目基本结构
 - 技术栈
- 44.项目分辨率响应式创建
 - flexible.js
 - cssrem插件
- 45.项目顶部信息条创建
- 46.页面主体创建
 - 大容器
 - 左中右
 - 左右图表展示区块容器样式
 - 左右每个区块内容插入容器槽口
 - 中间地图区域容器样式
- 47.图表前期准备
 - 全局设置Echarts与axios
 - Charts 全局引用
 - axios全局引用
- 48.后台接口创建express介绍
- 49.后台路由创建
- 50.api接口数据创建
- 50-1.解决跨域
- 51.图表1基本设置销售总量
 - 设置axios请求
 - 设置请求基准路径
- 52.处理数据
 - 动态展示图表
 - 添加echarts
- 53.图表一样式修改
 - 柱状图圆角与线段渐变色设置
 - 柱状图的柱状的位置与上面显示文字
- 54.图表2 地图展示
 - 获取地图数据
 - 设置地图
- 55.设置地图样式
- 56.在地图上设置散点标记图
- 57.设置提示框组件的视觉映射效果 (地图左下角效果)
 - 设置标题
- 58.图表3 产品库存统计分析图
 - 获取数据
 - 动态生成图表
- 59.类别分析图样式修改
- 60.图表4 产品月销图
 - 数据获取
 - 动态生成基本折线图
- 61.折线图样式设置
- 62.优化
- 63.图表5 产品库存统计图
 - 基本柱状图
- 64.完成堆叠效果
- 65.样式优化

- 66.项目打包
- 67.服务器购买与连接
- 68.nginx服务器使用
 - 代理服务器
 - 注意
 - 使用
 - 使用小扩展
- 69.项目运行
- 70.后端上线

课程介绍

讲师：千锋教育——奚皓源

- 公众号：大前端私房菜
- 回复：大数据可视化
- 获取大数据可视化Vue3.0与Echarts相关资料



Vue3.0与Echarts5.0

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”。

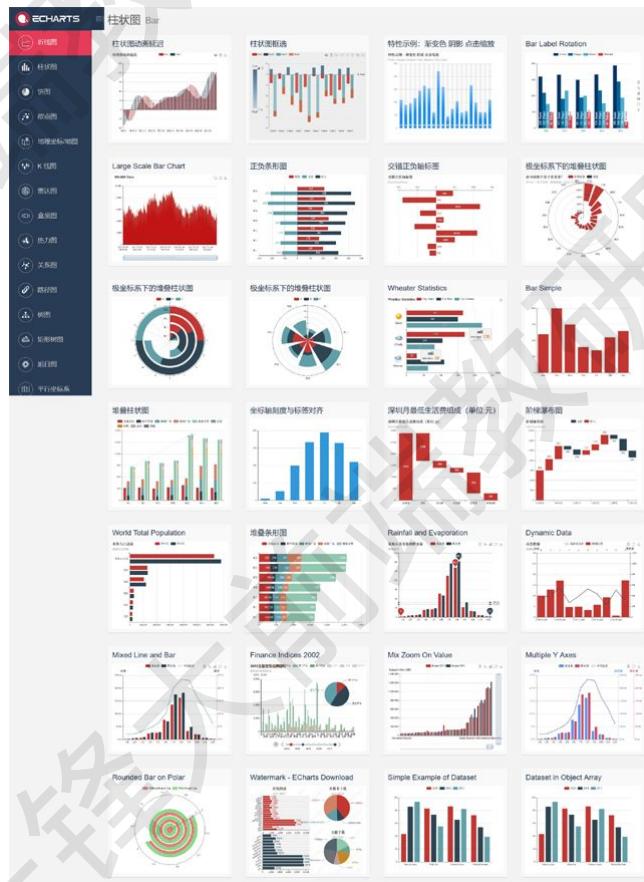
什么是数据可视化？

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

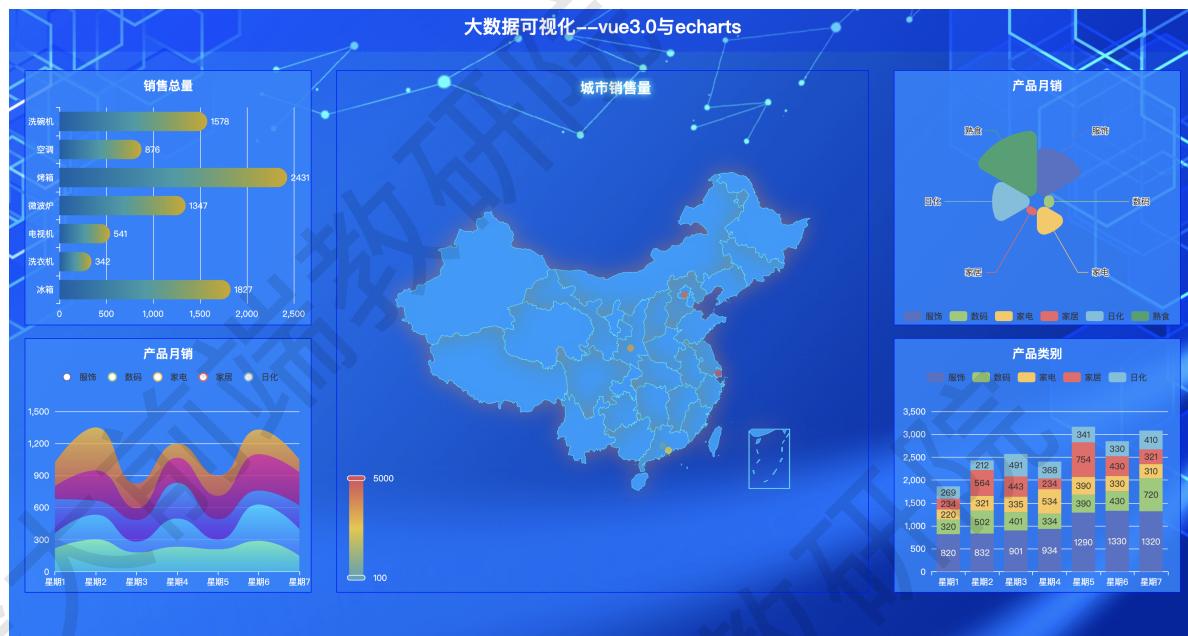
- 1、**丰富的可视化类型:** 提供了常规的折线图、柱状图、散点图、饼图、K线图，用于统计的盒形图，用于地理数据可视化的地图、热力图、线图，用于关系数据可视化的关系图、treemap、旭日图，多维数据可视化的平行坐标，还有用于 BI 的漏斗图，仪表盘，并且支持图与图之间的混搭。
- 2、**多种数据格式无需转换直接使用:** 内置的 dataset 属性 (4.0+) 支持直接传入包括二维表，key-value 等多种格式的数据源，此外还支持输入 TypedArray 格式的数据。
- 3、**千万数据的前端展现:** 通过增量渲染技术 (4.0+)，配合各种细致的优化，ECharts 能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选、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. HelloWorld 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 (选接)

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

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

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...', //字体
          // fontSize: 12, //大小
          // fontStyle: 'normal', //风格
          // fontWeight: 'normal', //粗细
          // subtextStyle: { //副标题文本样式{"color": "#aaa"} //字体
          // fontFamily: 'Arial, Verdana, sans...', //字体
          // fontSize: 12, //大小
          // fontStyle: 'normal', //风格
          // fontWeight: 'normal', //粗细
          // },
          // 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。
            }
        });
    }
}
</script>

```

```

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: "#ffffdd", // 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。主要是用来反应用对数据之间的关系,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
      }
    };
    myChart.setOption(option);
  }
}
</script>

```

```

        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

```

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

```

        },
        },
        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>

<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

```
<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

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

```

```

        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: {},
    }
  }
}
</script>

```

```
        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 },
          ],
        },
        {
          indicator: [
            {
              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, //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
        }
      ]
    }
  }
}
</script>
```

```

        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 myCharts=echarts.init(this.$refs.myChart)
    let options={
      series:[
        {
          type:"gauge",
          data:[
            {
              value:45,
              name:"提示信息"
            }
          ],
          detail:{
            valueAnimation:true
          },
          progress:{
            show:true
          }
        }
      ]
    }
    myCharts.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>

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

```

<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 data=[150, 230, 224, 218, 135, 147, 260]
let option = {
  xAxis: {
    type: 'category',
    data: xData
  },
  yAxis: {
    type: 'value'
  },
  series: [
    {
      data,
      type: 'line',
      smooth: true,
      markPoint: {
        data: [
          { type: 'max', name: 'Max' },
          { type: 'min', name: 'Min' }
        ]
      },
      markLine: {
        data: [{ type: 'average', name: 'Avg' }]
      }
    },
  ],
  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
    {
      type: 'slider',//滑动条型数据区域缩放组件
      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,
    },
],
};

myEcharts.setOption(options);
},
};

</script>

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

```

30方向切换树形图

```

<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",  
      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,
      },
      ],
    ];
  };

  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。
      }
    }
  };
}</script>
```

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

```

        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
          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);
}
},

```

中心位置

```

</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.demo);
    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>
```

41.vue3.0项目创建

1.电脑上安装node.js

网址: <https://nodejs.org/zh-cn/>

下载自己对应操作系统版本 安装即可

2.全局下载项目脚手架

打开cmd 输入 npm install -g @vue/cli

3. 创建项目

把cmd的路径切换到指定路径下 vue create 项目名

(3-1)选择项目配置模板 **选择第三项**自主选择你项目所需的配置

```
Please pick a preset:  
Default ([vue 2] babel, eslint) vue cli 2 默认的项目模板  
Default (Vue 3 Preview) ([Vue 3] babel, eslint) vue cli 3 默认的项目模板  
❯ Manually select features
```

(3-2)选择项目配置选项 勾选所需要的模块

```
? Check the features needed for your project:  
● Choose Vue version  
● Babel  
○ TypeScript  
○ Progressive Web App (PWA) Support  
● Router  
● Vuex  
❯● CSS Pre-processors  
● Linter / Formatter  
○ Unit Testing  
○ E2E Testing
```

(3-3)选择想要开始项目的Vue.js版本 选择 3.x

```
? Choose a version of vue.js that you want to start the project with  
2.x  
❯ 3.x (Preview)
```

(3-4)是否用history模式来创建路由 直接回车默认项目

```
? Use history mode for router? (Requires proper server setup for index fallback  
in pr  
oduction) (Y/n)
```

(3-5)选择CSS 预处理类型 选择LESS

```
? Pick a CSS pre-processor (PostCSS, Autoprefixer and CSS Modules are supported  
by default):  
Sass/SCSS (with dart-sass)  
Sass/SCSS (with node-sass)  
❯ Less  
Stylus
```

(3-6) 选择代码校验会犯 选择第一项 只进行报错提醒

```
? Pick a linter / formatter config: (Use arrow keys)
> ESLint with error prevention only
  ESLint + Airbnb config
  ESLint + Standard config
  ESLint + Prettier
```

(3-7)询问项目的什么时候校验格式(第一个是保存时, 第二个是提交时)

```
? Pick additional lint features: (Press <space> to select, <a> to toggle all, <i>
to
invert selection)
>❶ Lint on save
  ⚡ Lint and fix on commit
```

(3-8)询问项目的配置文件放在那里 (1.独立文件 2.package.json中)

```
? Where do you prefer placing config for Babel, ESLint, etc.? (Use arrow keys)
> In dedicated config files
  In package.json
```

(3-9)是否保存配置当做后续项目的可选配置 我们选择不保存

```
? Save this as a preset for future projects? (y/N)
```

4运行项目

把cmd的路径切换到你项目名下

npm run serve 启动项目

42.项目初始化?

- 1.删除src文件夹下的components文件夹下的HelloWorld.vue文件
- 2.删除views下的两个.vue文件
- 3.在views中新建我们的页面文件 homePage.vue文件

```
<template>
<div>
  我是页面
</div>
</template>

<script>
export default {

}

</script>

<style>

</style>
```

4.修改router下的index.js配置路由文件

```
import { createRouter, createWebHistory } from 'vue-router'

const routes = [
  {
    path: '/page',
    name: 'About',
    component: () => import('../views/homePage.vue')
  },
  // 设置路由重定向
  {
    path: '/',
    redirect: '/page'
  }
]

const router = createRouter({
  history: createWebHistory(process.env.BASE_URL),
  routes
})

export default router
```

5修改根组件默认显示内容与初始化项目样式

```
<template>
  <router-view/>
</template>

<style lang="less">
  *{
    margin: 0px;
    padding: 0px;
    box-sizing: border-box; //border-box 告诉浏览器去理解你设置的边框和内边距的值是包含在width内的。
  }
</style>
```

到此为止项目已经初始化完毕

4.3项目分辨率响应式分析与实施

项目基本结构

整体轮廓分为上下结构



在下半部分区域分为左中右结构



技术栈

1. vue3.0+vue-router4.0+axios
2. flex布局
3. LESS
4. rem屏幕适配
5. echarts5.0

44.项目分辨率响应式创建

我们的项目是需要根据页面的大小改变 做出响应式改变的 所以我们可以使用

我们可以使用 第三方插件flexible.js帮助我们修改html根节点的font-size大小 从而控制当前页面的rem(会根据页面的html根节点font-size大小改变而改变)样式改变

flexible.js

flexible.js web自适应方案 阿里团队开源的一个库。使用[flexible.js](#)轻松搞定各种不同的移动端设备兼容自适应问题。

1.下载 npm i -S lib-flexible

2.在main.js中进行配置

```
import { createApp } from 'vue'
import App from './App.vue'
import router from './router'
import store from './store'
// 引用
import 'lib-flexible/flexible.js'

createApp(App).use(store).use(router).mount('#app')
```

3.修改flexible配置

因为默认情况下只会在540px分辨率一下生效 所以我们需要根据我们的项目分辨率进行调整

在node_module/lib-flexible/flexible.js中修改代码如下

```
// 修改原始的
// if (width / dpr > 540) {
//   width = 540 * dpr;
// }
// var rem = width / 10;

// 修改成为
// 最小400px，最大适配2560px
if (width / dpr < 400) {
  width = 400 * dpr;
} else if (width / dpr > 2560) {
  width = 2560 * dpr;
}
// 设置成24等份，设计稿时1920px的，这样1rem就是80px（1920/24=80）方便使用
var rem = width / 24;
```

这个时候重启项目大家打开浏览器调试器 即可发现在浏览器大小改变的时候 在html根节点上会自动设置一个font-size

cssrem插件

我们在写代码的时候发现如果我们都根据80px为1rem在编写代码的时候转换非常的麻烦 所以我们可以 在vscode中安装一个cssrem的插件帮助我们进行转换 这样一来开发过程中会更加的方便

配置方式

在vscode扩展中找到 cssrem插件 最新名字叫px to rem & rpx 安装到vscode中 点击右下角设置

修改Root Font Size (基准font-size) 配置项为80即可



测试与使用

在写css的时候就会出现相应的rem转换结果

45.项目顶部信息条创建

1.设置背景图

把图片方法assets文件夹中 在app.vue中设置背景图

```
body{  
    background: url('~@/assets/bg.jpg') top center no-repeat;  
}
```

2.设置标题文字

```
<template>  
  <div>  
    <header>  
      <h1>大数据可视化--vue3.0与echarts</h1>  
    </header>  
  </div>  
</template>  
  
<script>  
export default {  
  
}  
</script>  
  
<style lang="less">  
header{  
  height: 1rem;  
  width: 100%;  
  /* 设置一个半透明淡蓝色 */  
  background-color: rgba(0, 0, 255, .2);  
  /* 把标题文字样式设置 */  
  h1{  
    font-size: .375rem;  
    color:#fff;  
    text-align: center;
```

```
        line-height: 1rem;
    }
}
</style>
```

46. 页面主体创建

主体部分是下面的左中右



大容器

1. 创建一个大容器来容纳绿色 红色 黄色三个区域

在 homepage.vue 页面中创建一个大容器

```
<template>
  <div>
    <header>
      <h1>大数据可视化--vue3.0与echarts</h1>
    </header>
    <!-- 大容器 -->
    <section class="container">

    </section>
  </div>
</template>
```

创建容器样式

```
<style lang="less">
  header{
    height: 1rem;
    width: 100%;
    /* 设置一个半透明淡蓝色 */
    background-color: rgba(0, 0, 255, .2);
    /* 把标题文字样式设置 */
    h1{
      font-size: .375rem;
      color:#fff;
      text-align: center;
      line-height: 1rem;
    }
  }
  // 主体容器样式
  .container{
    // 这里就不需要设置使用rem了 使用rem那么页面就会根据html根结点大小改变而改变了
    min-width: 1200px;
    max-width: 2048px;
    margin: 0 auto;
    // 盒子上10px 左右10px 下0的外边距
    padding: .125rem .125rem 0;
    // 测试完成看到样式就删除掉
    height: 500px;
    background-color: gray;
  }
</style>
```

```
</style>
```

左中右

接下来我们可以创建左中右这三个部分。那么他们的占比分别是3 5 3 这个时候我们可以使用flex布局来分割他们所占的区块大小



1. 创建左中右三个页面容器

```
<template>
<div>
  <header>
    <h1>大数据可视化--vue3.0与echarts</h1>
  </header>
  <!-- 大容器 -->
  <section class='container'>
    <!-- 左容器 -->
    <section class='itemLeft'>1</section>
    <!-- 中容器 -->
    <section class='itemCenter'>2</section>
    <!-- 右容器 -->
    <section class='itemRight'>3</section>
  </section>
</div>
</template>
```

2. 设置样式

```
<style lang="less">
  header{
    height: 1rem;
    width: 100%;
    /* 设置一个半透明淡蓝色 */
    background-color: rgba(0, 0, 255, .2);
    /* 把标题文字样式设置 */
    h1{
      font-size: .375rem;
      color:#fff;
      text-align: center;
      line-height: 1rem;
    }
  }
  // 主体容器样式
  .container{
    // 这里就不需要设置使用rem了
    min-width: 1200px;
    max-width: 2048px;
    margin: 0 auto;
    // 盒子上10px 左右10px 下0的外边距
    padding: .125rem .125rem 0;
    display: flex; //父容器设置flex布局才能在子元素使用

    // 设置左中右的占比 但是不要忘了在父容器要设置flex
    .itemLeft,.itemRight{
      flex: 3;
```

```
        }
    .itemConter{
        flex: 5;
    }

}
</style>
```

运行之后会发现 页面的左和右占比是页面各的3份。而中间是占比5份

左右图表展示区块容器样式

大家会发现我们要展示的4个区域的容器效果是一样的。所以我们可以剥离成一个组件 然后重复调用即可。并且在其中放置slot槽口 后期方便向容器内插入图表



创建容器组件

在components文件夹下创建 itemPage.vue

```
<template>
<div>
    容器组件
</div>
</template>

<script>
export default {

}
</script>

<style>
</style>
```

编写样式与插槽

```
<template>
<div class='item'>
    <!-- 设置插槽 -->
    <slot/>
</div>
</template>

<script>
export default {

}
</script>

<style>
.item{
    /* 高度410px */
    height: 5.125rem;
    border: 1px solid blue;
```

```

/* 外边距20px */
margin: .25rem;
background-color: rgba(13, 130, 255, 0.851);
}
</style>

```

在views下的homePage中引用调用使用

```

<template>
<div>
  <header>
    <h1>大数据可视化--vue3.0与echarts</h1>
  </header>
  <!-- 大容器 -->
  <section class='container'>
    <!-- 左容器 -->
    <section class='itemLeft'>
      <!-- 使用组件 -->
      <ItemPage/>
      <ItemPage/>
    </section>
    <!-- 中容器 -->
    <section class='itemCenter'>2</section>
    <!-- 右容器 -->
    <section class='itemRight'>
      <!-- 使用组件 -->
      <ItemPage/>
      <ItemPage/>
    </section>
  </section>
</div>
</template>

<script>
// 引用组件
import ItemPage from "@/components/itemPage.vue"
export default {
  components: {
    ItemPage
  }
}
</script>

```

运行之后大家会发现左右区块就展现出4个容器

左右每个区块内容插入容器槽口

我们一共4个图标 使用一个公共的组件容器 所以我们编写这4个不同图表的组件并且 分别显示



1.创建4个组件 在components下 itemOne.vue等等 一共4个

然后在4个文件中分别设置相关内容与样式 (每个图表的标题不一样要修改)

```

<template>
<div>

```

```
<h2>图表1</h2>
<div class="chart">
    容纳后期的图表
</div>
</div>
</template>

<script>
export default {

}

</script>

<style scoped>
h2{
    /* 48像素 */
    height: 0.6rem;
    color: #fff;
    line-height: 0.6rem;
    text-align: center;
    font-size: 0.25rem;
}

.chart{
    /* 高度360 */
    height: 4.5rem;
    background-color: gray;
}
</style>
```

在homePage.vue中引用调用使用这4个组件

```
<template>
<div>
    <header>
        <h1>大数据可视化--vue3.0与echarts</h1>
    </header>
    <!-- 大容器 -->
    <section class='container'>
        <!-- 左容器 -->
        <section class='itemLeft'>
            <!-- 使用组件 -->
            <ItemPage><itemOne/></ItemPage>
            <ItemPage><itemTwo/></ItemPage>
        </section>
        <!-- 中容器 -->
        <section class='itemCenter'>2</section>
        <!-- 右容器 -->
        <section class='itemRight'>
            <!-- 使用组件 -->
            <ItemPage><itemThree/></ItemPage>
            <ItemPage><itemFour/></ItemPage>
        </section>
    </section>
</div>
</template>
```

```
<script>
// 引用组件
import ItemPage from "@/components/itemPage.vue"

// 左右4个小组件的引用
import itemOne from "@/components/itemOne.vue"
import itemTwo from "@/components/itemTwo.vue"
import itemThree from "@/components/itemThree.vue"
import itemFour from "@/components/itemFour.vue"

export default {
  components:{
    ItemPage,itemOne,itemTwo,itemThree,itemFour
  }
}
</script>
```

中间地图区域容器样式

在views文件夹下的 homePage.vue 中设置中间区域容器样式

```
.itemCenter{
  // 高度840px
  height: 10.5rem;
  border: 1px solid blue;
  // 内边距10px
  padding: 0.125rem;
  // 外边距20px
  margin: 0.25rem;
}
```

47.图表前期准备

全局设置Echarts与axios

Charts 全局引用

1.下载 npm install --save echarts

2.0的写法

在vue2.0中使用如下写法吧echarts挂载在vue实例上 但是这招在3.0行不通了

在main.js中进行引用和调用

```
import { createApp } from 'vue'
import App from './App.vue'
import router from './router'
import store from './store'
// 引用
import 'lib-flexible/flexible.js'
// 引用echarts
import * as echarts from "echarts"
Vue.prototype.$echarts=echarts;

createApp(App).use(store).use(router).mount('#app')
```

vue3中使用Provide/Inject依赖注入，将替代vue2中在原型链上挂载一些属性

在app.vue中使用provider来给后代们提供数据

```
<script>
// 1.引用proivde
import {provide} from "vue"
// 2.引用echarts
import * as echarts from "echarts"
export default {
  setup() {
    provide("echarts",echarts)//第一个参数是名字 第二个参数是你传递的内容
  },
}
</script>
```

在想使用的组件中使用inject来接受

在views下的homePage.vue测试

```
// 引用inject
import {inject} from 'vue'
export default {
  components:{
    ItemPage,itemOne,itemTwo,itemThree,itemFour
  },
  setup(){
    // 测试使用echarts
    let $echarts= inject("echarts")
    console.log($echarts)
  }
}
```

大家在console中可以看到可以正常使用了

axios全局引用

axios使用于上面相同方式

1.下载 npm install --save axios

2.在app.vue中使用provider来给后代们提供数据

```
<script>
// 1.引用proivde
import {provide} from "vue"
// 2.引用echarts
import * as echarts from "echarts"
// 引用axios
import axios from 'axios'
export default {
  setup() {
    provide("echarts",echarts)//第一个参数是名字 第二个参数是你传递的内容
    provide("axios",axios)//第一个参数是名字 第二个参数是你传递的内容
  },
}
</script>
```

在想使用的组件中使用inject来接受

在views下的homePage.vue测试

```
// 引用inject
import {inject} from 'vue'
export default {
  components:{
    ItemPage,itemOne,itemTwo,itemThree,itemFour
  },
  setup(){
    // 测试使用echarts
    let $echarts= inject("echarts")
    let $http= inject("axios")
    console.log($echarts)
    console.log($http)
  }
}
```

48.后台接口创建express介绍

49.后台路由创建

1.创建一个文件夹 server 在其中创建index.js与router文件夹容来容纳代码

2.在router下创建4个文件分别容纳 对应的接口

```
// 存放路由的文件
var express=require("express");
var router=express.Router()
// 设置路由
router.get("/data",function(req,res){
    res.send({msg:"第一个接口"})
})

// 暴露路由
module.exports=router
```

3.在index.js下引用使用刚才创建的内容

```
var express=require("express");
var app=express();
// 引用路由文件
var chartOne=require("./router/one.js");
var chartTwo=require("./router/two.js");
var chartThree=require("./router/three.js");
var chartFour=require("./router/four.js");
// 中间件中使用路由
app.use("/one",chartOne)
app.use("/two",chartTwo)
app.use("/three",chartThree)
app.use("/four",chartFour)
// 请求是localhost:3000/user/路由文件中的地址
app.listen(3000)
```

50.api接口数据创建

1.在server文件夹下创建mock文件夹用来容纳数据(数据可以从代码中获取)

2.引用并且把数据返回给前台

```
var express=require("express");
var router=express.Router()
// 引用数据
let data=require("../mock/one.json")
router.get("/data",function(req,res){
    // 数据返回给前台
    res.send({msg:"第1个接口",data})
})

module.exports=router
```

50-1.解决跨域

```

var express=require("express");
var app=express();
app.use(function(req,res,next){
    res.header('Access-Control-Allow-Origin', '*');

    res.header('Access-Control-Allow-Headers', 'Content-Type,Content-Length,
Authorization, Accept, X-Requested-with , yourHeaderFeild');

    res.header('Access-Control-Allow-Methods', 'PUT, POST, GET,
DELETE,OPTIONS');
    // 千万不要网%%￥￥￥####
    // 千万不要网
    // 千万不要网
    next();
})

```

51.图表1基本设置销售总量

1.在components文件夹下的 itemOne.vue中 设置图表1

```

<template>
<div>
    <h2>图表1</h2>
    <div id="chartDom" class="chart">
        容纳后期的图表
    </div>
</div>
</template>

<script>
// 1引用inject获取全局echarts
import { inject, onMounted } from "vue"
export default {
    setup() {
        // 2得到echarts对象
        let $echarts= inject("echarts")
        //3.需要获取到element,所以是onMounted 别忘了上面引用
        onMounted(() => {
            // 4.初始化echarts 别忘了给上面echarts容器添加id
            let myChart = $echarts.init(document.getElementById("chartDom"));
            // 5.绘制图表
            myChart.setOption({
                ...
            })
        })
    }
}
</script>
<style scoped>
h2{
    /* 48像素 */
    height: 0.6rem;
    color: #fff;
    line-height: 0.6rem;
    text-align: center;
    font-size: 0.25rem;
}

```

```
        }
    .chart{
        /* 高度360 */
        height: 4.5rem;
        background-color: gray;
    }
</style>
```

2.我们要完成的横向柱状图



所以在设置echarts配置的时候。给xAxis 的type设置value数值轴。给yAxis设置 category类目轴

```
<template>
<div>
    <h2>图表1</h2>
    <div id="chartDom" class="chart">
        容纳后期的图表
    </div>
</div>
</template>

<script>
// 1引用inject获取全局echarts
import { inject, onMounted } from "vue"
export default {
    setup() {
        // 2得到echarts对象
        let $echarts= inject("echarts")
        //3.需要获取到element,所以是onMounted 别忘了上面引用
        onMounted(() => {
            // 4.初始化echarts 别忘了给上面echarts容器添加id
            let myChart = $echarts.init(document.getElementById("chartDom"));
            // 5.绘制图表
            myChart.setOption({
                xAxis:{
                    type:"value"
                },
                yAxis:[
                    type:"category"
                ],
                series:[
                    {
                        type:"bar"
                    }
                ]
            })
        })
    }
}
</script>
<style scoped>
h2{
    /* 48像素 */
}
```

```
height: 0.6rem;
color: #fff;
line-height: 0.6rem;
text-align: center;
font-size: 0.25rem;

}

.chart{
/* 高度360 */
height: 4.5rem;
background-color: gray;
}

</style>
```

3 接下来我们需要图表展示的数据(后台提供)

设置axios请求

在组件内容请求数据

```
<template>
<div>
<h2>图表1</h2>
<div id="chartDom" class="chart">
    容纳后期的图表
</div>
</div>
</template>

<script>
import { inject, onMounted } from "vue"
export default {
    setup() {

        // let $echarts= inject("echarts")
        // 引用axios
        let $http= inject("axios")
        // 设置数据请求方法 不要忘了return
        async function getState(){
            let chartData=await $http({url:"http://localhost:3000/one/data"})
            console.log(chartData)
        }

        onMounted(() => {

            // let myChart = $echarts.init(document.getElementById("chartDom"));
            // myChart.setOption({
            //     xAxis:{
            //         type:"value"
            //     },
            //     yAxios:{
            //         type:"category"
            //     },
            //     series:[
            //         {
            //             type:"bar"
            //         }
            //     ]
            // })
        })
    }
}
</script>
```

```

        //      ]
        // }]

        // 测试数据请求的数据 但是先要把上面的注释掉否则会报错
        getState()
    })

    return {
        getState
    }
}
}

</script>
<style scoped>
h2{
    /* 48像素 */
    height: 0.6rem;
    color: #fff;
    line-height: 0.6rem;
    text-align: center;
    font-size: 0.25rem;
}

.chart{
    /* 高度360 */
    height: 4.5rem;
    background-color: gray;
}
</style>

```

设置请求基准路径

后续我们有很多的请求 或者后期请求地址改变的时候我们在一个个组件的修改非常的麻烦 所以我们可以设置请求基准路径方便修改

1.在app.vue中设置

```

<script>
// 1.引用provide
import {provide} from "vue"
// 2.引用echarts
import * as echarts from "echarts"
// 引用axios
import axios from 'axios'
// 设置请求基准路径
axios.defaults.baseURL="http://localhost:3000/"

export default {
    setup() {
        provide("echarts",echarts)//第一个参数是名字 第二个参数是你传递的内容
        provide("axios",axios)//第一个参数是名字 第二个参数是你传递的内容
    },
}
</script>

```

2.在需要请求的地方只需要写请求的路由地址即可

```
async function getState(){
    let chartData=await $http({url:"chartDataOne"})
    console.log(chartData)
}
```

52.处理数据

我们请求来的数据事一个数组对象 那么我们需要把x轴的数据于y轴的数据取出来变成两个数组方便echarts使用

```
<template>
<div>
    <h2>图表1</h2>
    <div id="chartDom" class="chart">
        容纳后期的图表
    </div>
</div>
</template>

<script>
// 1.引用reactive
import {inject, onMounted, reactive} from "vue"
export default {
    setup() {

        // let $echarts= inject("echarts")
        let $http= inject("axios")
        async function getState(){
            // 3.吧请求过来的值传递给变量接收方便使用
            data=await $http({url:"chartDataOne"})

        }
        // 2.创建接受请求的数据于接受x轴的数据于y轴的数据 别忘了return
        let data=reactive({})
        let xdata=reactive([])
        let ydata=reactive([])
        // 4.创建处理数据的方法
        function setData(){
            console.log("data",data)
            // 便利数据并且去除对应的值
            xdata=data.data.data.chartData.map(v=>v.title)
            ydata=data.data.data.chartData.map(v=>v.num)
            console.log("x",xdata)
            console.log("y",ydata)
        }

        onMounted(() => {

            // let myChart = $echarts.init(document.getElementById("chartDom"));
            // myChart.setOption({
            //     xAxis:{
            //         type:"value"
            //     },
            //     yAxios:{
            //         type:"category"
            //     },
            // 
```

```

        //           series:[
        //           {
        //             type:"bar"
        //           }
        //         ]
        //   }

      // 5在请求成功之后调用处理数据的方法
      getState().then(()=>{
        // 测试下处理的数据
        setData()
      })

    })

  return {
    getState,data,xdata,ydata,setData
  }
}
}

</script>
<style scoped>
h2{
  /* 48像素 */
  height: 0.6rem;
  color: #fff;
  line-height: 0.6rem;
  text-align: center;
  font-size: 0.25rem;

}

.chart{
  /* 高度360 */
  height: 4.5rem;
  background-color: gray;
}
</style>

```

动态展示图表

```

<template>
<div>
  <h2>图表1</h2>
  <div id="chartDom" class="chart">
    容纳后期的图表
  </div>
</div>
</template>

<script>

import { inject, onMounted, reactive } from "vue"
export default {
  setup() {
    // 1. 打开echarts的注释
    let $echarts= inject("echarts")
  }
}

```

```
let $http= inject("axios")
async function getState(){
    data=await $http({url:"chartDataOne"})
}

let data=reactive({})
let xdata=reactive([])
let ydata=reactive([])
function setData(){
    console.log("data",data)
    xdata=data.data.data.chartData.map(v=>v.title)//名字在y轴展示
    ydata=data.data.data.chartData.map(v=>v.num)//数据在x轴展示
    console.log("x",xdata)
    console.log("y",ydata)
}

onMounted(() => {
    getState().then(()=>{
        setData()
        // 2.吧图表展示放到数据请求成功当中 并且传入对应数据
        let myChart = $echarts.init(document.getElementById("chartDom"));
        myChart.setOption({
            xAxis:{
                type:"value"
            },
            yAxis:{
                type:"category",
                // 2.插入数据
                data:xdata
            },
            series:[
                {
                    data:ydata,
                    type:"bar"
                }
            ]
        })
    })
}

return {
    getState,data,xdata,ydata,setData
}
}

```

```
</script>
<style scoped>
h2{
    /* 48像素 */
    height: 0.6rem;
    color: #fff;
    line-height: 0.6rem;
    text-align: center;
    font-size: 0.25rem;
}
```

```
}

.chart{
    /* 高度360 */
    height: 4.5rem;
    background-color: gray;
}

</style>
```

添加echarts

echart图表本身是提供了一个 `resize` 的函数的。是当浏览器发生 `resize` 事件的时候，让其触发echart的 `resize` 事件，重绘canvas。

用 `window.onresize = myChart.resize;` 可以完成自适应，就是把 `window` 的 `onresize` 事件赋值为echart的 `resize` 事件

```
<template>
<div>
    <h2>图表1</h2>
    <div id="chartDom" class="chart">
        容纳后期的图表
    </div>
</div>
</template>

<script>

import { inject, onMounted, reactive } from "vue"
export default {
    setup() {
        let $echarts= inject("echarts")
        let $http= inject("axios")
        async function getState(){
            data=await $http({url:"chartDataOne"})
        }
        let data=reactive({})
        let xdata=reactive([])
        let ydata=reactive([])
        function setData(){
            console.log("data",data)
            xdata=data.data.data.chartData.map(v=>v.title)
            ydata=data.data.data.chartData.map(v=>v.num)
            console.log("x",xdata)
            console.log("y",ydata)
        }
        onMounted(() => {
            getState().then(()=>{
                setData()
                let myChart = $echarts.init(document.getElementById("chartDom"));
                myChart.setOption({

```

```

        xAixs: {
            type: "value"
        },
        yAixs: {
            type: "category",
            data: xdata
        },
        series: [
            {
                data: ydata,
                type: "bar"
            }
        ]
    })
    // 让echarts根据浏览器大小改变动态该改变
    window.onresize = function () { //自适应大小
        myChart.resize();
    };
}

return {
    getState, data, xdata, ydata, setData
}
}
}

```

</script>

```

<style scoped>
    h2{
        /* 48像素 */
        height: 0.6rem;
        color: #fff;
        line-height: 0.6rem;
        text-align: center;
        font-size: 0.25rem;
    }
    .chart{
        /* 高度360 */
        height: 4.5rem;
        background-color: gray;
    }
</style>

```

53. 图表一样式修改

柱状图圆角与线段渐变色设置

在components下的itemOne.vue

```

<template>
<div>
    <h2>图表1</h2>

```

```
<div id="chartDom" class="chart">
    容纳后期的图表
</div>
</div>
</template>

<script>

import { inject, onMounted, reactive } from "vue"
export default {
    setup() {
        let $echarts= inject("echarts")
        let $http= inject("axios")
        async function getState(){
            data=await $http({url:"chartDataOne"})
        }
        let data=reactive({})
        let xdata=reactive([])
        let ydata=reactive([])
        function setData(){
            console.log("data",data)
            xdata=data.data.data.chartData.map(v=>v.title)
            ydata=data.data.data.chartData.map(v=>v.num)
            console.log("x",xdata)
            console.log("y",ydata)
        }
    }

    onMounted(() => {

        getState().then(()=>{
            setData()
            let myChart = $echarts.init(document.getElementById("chartDom"));
            myChart.setOption({
                xAxis:{
                    type:"value"
                },
                yAxis:{
                    type:"category",
                    data:xdata
                },
                series:[
                    {
                        data:ydata,
                        type:"bar",
                        // 设置柱状图区域样式
                        itemStyle: {
                            normal: {
                                // 如果传递数组，则需要传递四个元素，依次表示 左上，右上，
                                // 右下、左下
                                barBorderRadius: [0,20,20,0],
                                // color:"red"如果写死值那么是一个固定的颜色
                                // 我们需要设置渐变色使用
                                // new
                                echarts.graphic.LinearGradient(a,b,c,d,arr)来进行设置
                                // a ,b,c,d为0, 1
                            }
                        }
                    }
                ]
            })
        })
    })
}

```

```
// a:1 arr中的颜色右到左  
// c:1 arr中的颜色左到右  
// b:1 arr中的颜色下到上  
// d:1 arr中的颜色上到下  
color:new  
$echarts.graphic.LinearGradient(0,0,1,0,[  
    {  
        offset:0,  
        color:'#005eaa'  
    },  
    {  
        offset:0.5,  
        color:'#339ca8'  
    },  
    {  
        offset:1,  
        color:'#cda819'  
    }  
])  
}  
]  
}  
})  
}  
  
window.onresize = function () {  
    myChart.resize();  
};  
  
})  
}  
  
return {  
    getState,data,xdata,ydata,setData  
}  
}  
}  
}  
}  
</script>  
<style scoped>  
h2{  
    /* 48像素 */  
    height: 0.6rem;  
    color: #fff;  
    line-height: 0.6rem;  
    text-align: center;  
    font-size: 0.25rem;  
  
}  
.chart{  
    /* 高度360 */  
    height: 4.5rem;  
    background-color: gray;  
}  
</style>
```

柱状图的柱状的位置与上面显示文字

```
<template>
<div>
    <h2>图表1</h2>
    <div id="chartDom" class="chart">
        容纳后期的图表
    </div>
</div>
</template>

<script>

import { inject, onMounted, reactive } from "vue"
export default {
    setup() {
        let $echarts= inject("echarts")
        let $http= inject("axios")
        async function getState(){
            data=await $http({url:"chartDataOne"})
        }
        let data=reactive({})
        let xdata=reactive([])
        let ydata=reactive([])
        function setData(){
            console.log("data",data)
            xdata=data.data.data.chartData.map(v=>v.title)
            ydata=data.data.data.chartData.map(v=>v.num)
            console.log("x",xdata)
            console.log("y",ydata)
        }
        onMounted(() => {
            getState().then(()=>{
                setData()
                let myChart = $echarts.init(document.getElementById("chartDom"));
                myChart.setOption({
                    xAxis:{
                        // 设置坐标轴上文字颜色
                    axisLine:{
                        linestyle:{
                            color:"#fff"
                        }
                    },
                    type:"value"
                },
                yAxis:{
                    // 设置坐标轴上文字颜色
                    axisLine:{
                        linestyle:{
                            color:"#fff"
                        }
                    },
                }
            })
        })
    }
}


```

```
        type:"category",
        data:xdata
    },
    // 配置图标的位置 不包含坐标轴中的文字
    grid:{
        top:'3%',
        left:'1%',
        bottom:'3%',
        right:'6%',
        containLabel:true // 包含坐标轴中的文字
    },
    series:[
        {
            data:ydata,
            type:"bar",
            // 设置图形上的文字
            label:{
                show:true, // 显示数值
                position:'right',//位置
                textStyle:{
                    color:'#fff'
                }
            },
            itemStyle: {
                normal: {
                    barBorderRadius: [0,20,20,0],
                    // 我们需要设置渐变色使用
                    // new
                    echarts.graphic.LinearGradient(a,b,c,d,arr)来进行设置
                    // a ,b,c,d为0, 1
                    // a:1 arr中的颜色右到左
                    // c:1 arr中的颜色左到右
                    // b:1 arr中的颜色下到上
                    // d:1 arr中的颜色上到下
                    color:new
                }
            }
        }
    ],
    $echarts.graphic.LinearGradient(0,0,1,0,[
        {
            offset:0,
            color:'#005eaa'
        },
        {
            offset:0.5,
            color:'#339ca8'
        },
        {
            offset:1,
            color:'#cda819'
        }
    ])
},
],
}
})
```

```
        }
    })

    return {
        getState,data,xdata,ydata,setData
    }
}

}
</script>
<style scoped>
h2{
    /* 48像素 */
    height: 0.6rem;
    color: #fff;
    line-height: 0.6rem;
    text-align: center;
    font-size: 0.25rem;
}

.chart{
    /* 高度360 */
    height: 4.5rem;
    /* background-color: gray; */
}
</style>
```

54.图表2 地图展示

如果要展示地图那么需要中国地图的矢量数据与省份数据 在public中的map文件夹已经提供了
由于数据实在我们本地 所以我们在启动项目的时候可以直接在浏览器上输入

<http://localhost:8080/map/china.json>即可看到数据

我们在components文件夹下创建一个mapPage.vue组件用来容纳地图。同时在views下的homePage.vue中引用调用并且在页面中间的div中使用

```
<template>
<div class="map">
    map
</div>
</template>

<script>
export default {

}
</script>

<style lang="less">
.map{
    width: 100%;
    height: 100%;
}
</style>
```

获取地图数据

因为我们在项目中app.vue中 设置了 请求基准路径

```
axios.defaults.baseURL="http://localhost:3000/"
```

所以我们不能在使用全局的axios 否则会使用基准路径完成请求

我们单独引用axios进行数据的请求

```
<template>
  <div class="map">
    map
  </div>
</template>

<script>
// 1.引用
import axios from "axios"
import {onMounted, reactive} from "vue"
export default {
  setup(){
    // 2.设置请求 并且创建变量接收获取来的数据 不要忘了报漏
    let mapData=reactive({})
    async function getState(){
      mapData=await axios.get("http://localhost:8080/map/china.json")
    }

    onMounted(()=>{
      // 3发送请求
      getState().then(()=>{
        console.log("map",mapData)
      })
    })
    return {
      mapData
    }
  }
}
</script>

<style lang="less">
.map{
  width: 100%;
  height: 100%;
}
</style>
```

当然我们也可以直接引用数据不用请求

```
import china from 'map/json/china.json' //本地路径
```

就可以不用上面的请求使用

设置地图

echarts.registerMap (名字, 数据)

```
<template>
  <div class="map" id='map'>
    map
  </div>
</template>

<script>

import axios from "axios"
// 1.引用
import {onMounted, reactive, inject} from "vue"
export default {
  setup(){
    // 2.得到echarts
    let $echarts= inject("echarts")
    let mapData=reactive({})
    async function getState(){

      mapData=await axios.get("http://localhost:8080/map/china.json")
    }

    onMounted(()=>{
      getState().then(()=>{
        console.log("map",mapData)
        // 3.设置地图
        $echarts.registerMap('china', mapData.data);
        var chart = $echarts.init(document.getElementById('map'));
        chart.setOption({
          geo: {
            map: 'china'
          }
        });
        return {
          mapData
        }
      })
    })
  }
}
</script>

<style lang="less">
.map{
  width: 100%;
  height: 100%;
}
</style>
```

55.设置地图样式

```
<template>
  <div class="map" id='map'>
```

```

    map
  </div>
</template>

<script>

import axios from "axios"
import {onMounted, reactive, inject} from "vue"
export default {
  setup(){
    let $echarts= inject("echarts")
    let mapData=reactive({})
    async function getState(){

      mapData=await axios.get("http://localhost:8080/map/china.json")
    }

    onMounted(()=>{
      getState().then(()=>{
        console.log("map",mapData)
        $echarts.registerMap('china', mapData.data);
        var chart = $echarts.init(document.getElementById('map'));
        chart.setOption({
          geo: {
            map: 'china',
            itemStyle:{//地图区域的多边形 图形样式
              areaColor:"#0099ff",//地图区域的颜色。
              borderColor:"#00ffff",//图形的描边颜色。
              shadowColor: 'rgba(230,130, 70, 0.5)',//橙色
              shadowBlur: 30,//图形阴影的模糊大小
              emphasis: {}//高亮状态下的多边形和标签样式。
            }
          }
        })
      return {
        mapData
      }
    })
  }
}</script>

```

56.在地图上设置散点标记图

```

<template>
  <div class="map" id="map">map</div>
</template>

<script>
import axios from "axios";
import { onMounted, reactive, inject } from "vue";
export default {

```

```
setup() {
  let $echarts = inject("echarts");
  let mapData = reactive({});

  async function getState() {
    mapData = await axios.get("http://localhost:8080/map/china.json");
  }

  onMounted(() => {
    getState().then(() => {
      console.log("map", mapData);
      $echarts.registerMap("china", mapData.data);
      var chart = $echarts.init(document.getElementById("map"));
      chart.setOption({
        geo: {
          map: "china",
          itemStyle: {
            //地图区域的多边形 图形样式
            areaColor: "#0099ff", //地图区域的颜色。
            borderColor: "#00ffff", //图形的描边颜色。
            shadowColor: "rgba(230,130, 70, 0.5)", //橙色
            shadowBlur: 30, //图形阴影的模糊大小
            emphasis: {
              //高亮状态下的多边形和标签样式。
              focus: "self", //在高亮图形时，是否淡出其它数据的图形已达到聚焦的效果
              self: 'self' //只聚焦（不淡出）当前高亮的数据的图形。
            },
            },
          },
        series: [
          {
            type: "scatter", //类型散点图
            itemStyle: { //散点图的颜色
              color: "red",
            },
            name: "所在城市销售额",
            coordinateSystem: "geo", //该系列使用的坐标系 geo使用地理坐标系
            data: [
              { name: "北京", value: [116.46, 39.92, 4367] },
              { name: "上海", value: [121.48, 31.22, 8675] },
              { name: "深圳", value: [114.07, 22.62, 2461] },
              { name: "广州", value: [113.23, 23.16, 187] },
              { name: "西安", value: [108.45, 34, 3421] },
            ],
          },
          ],
        },
        // tooltip: {
        //   trigger: "item",
        // },
        // visualMap: {
        //   type: "continuous", // 连续型
        //   min: 100, // 值域最小值，必须参数
        //   max: 5000, // 值域最大值，必须参数
        //   calculable: true, // 是否启用滑动空间
        //   inRange: {
        //     color: ["#50a3ba", "#eac736", "#d94e5d"], // 指定数值从低到高时的颜色变化
        //   },
        //   textStyle: {

```

```

        //      color: "#fff", // 值域控件的文本颜色
        //    },
        //  },
        //);
      });
    });
  return {
    mapData,
  };
},
};

</script>

<style lang="less">
.map {
  width: 100%;
  height: 100%;
}
</style>

```

57.设置提示框组件的视觉映射效果（地图左下角效果）

```

<template>
  <div class="map" id="map">map</div>
</template>

<script>
import axios from "axios";
import { onMounted, reactive, inject } from "vue";
export default {
  setup() {
    let $echarts = inject("echarts");
    let mapData = reactive({});

    async function getState() {
      mapData = await axios.get("http://localhost:8080/map/china.json");
    }

    onMounted(() => {
      getState().then(() => {
        console.log("map", mapData);
        $echarts.registerMap("china", mapData.data);
        var chart = $echarts.init(document.getElementById("map"));
        chart.setOption({
          geo: {
            map: "china",
            itemStyle: {
              areaColor: "#0099ff",
              borderColor: "#00ffff",
              shadowColor: "rgba(230,130, 70, 0.5)",
              shadowBlur: 30,
              emphasis: {
                focus: "self",
              },
            },
          },
          series: [
            {

```

```
type: "scatter",
itemStyle: {
    color:"red",
},
name: "所在城市销售额",
coordinateSystem: "geo",
data: [
    { name: "北京", value: [116.46, 39.92, 4367] },
    { name: "上海", value: [121.48, 31.22, 8675] },
    { name: "深圳", value: [114.07, 22.62, 2461] },
    { name: "广州", value: [113.23, 23.16, 187] },
    { name: "西安", value: [108.45, 34, 3421] },
],
],
],
tooltip: {//提示框组件。
trigger: "item",
},
visualMap: {//是视觉映射组件就是地图左下角的选择器
type: "continuous", // 连续型
min: 100, // 值域最小值, 必须参数
max: 5000, // 值域最大值, 必须参数
calculable: true, // 是否启用滑动空间
inRange: {
color: ["#50a3ba", "#eac736", "#d94e5d"], // 指定数值从低到高时的颜色变化
},
textStyle: {
color: "#fff", // 值域控件的文本颜色
},
},
},
);
});
return {
mapData,
};
},
);
</script>

<style lang="less">
.map {
width: 100%;
height: 100%;
}
</style>
```

设置标题

```
<template>
<div class="map" id="map">map</div>
</template>

<script>
```

```
import axios from "axios";
import { onMounted, reactive, inject } from "vue";
export default {
  setup() {
    let $echarts = inject("echarts");
    let mapData = reactive({});
    async function getState() {
      mapData = await axios.get("http://localhost:8080/map/china.json");
    }
  }

  onMounted(() => {
    getState().then(() => {
      console.log("map", mapData);
      $echarts.registerMap("china", mapData.data);
      var chart = $echarts.init(document.getElementById("map"));
      chart.setOption({
        title:{
          text:"城市销售量",
          left:"45%",
          textStyle:{
            color:"#fff",
            fontsize:20,
            textShadowBlur:10,//文字本身的阴影长度。
            textShadowColor :"#33ffff",
          }
        },
        geo: {
          map: "china",
          itemStyle: {
            areaColor: "#0099ff",
            borderColor: "#00ffff",
            shadowColor: "rgba(230,130, 70, 0.5)",
            shadowBlur: 30,
            emphasis: {
              focus: "self",
            },
          },
        },
        series: [
          {
            type: "scatter",
            itemStyle: {
              color:"red",
            },
            name: "所在城市销售额",
            coordinateSystem: "geo",
            data: [
              { name: "北京", value: [116.46, 39.92, 4367] },
              { name: "上海", value: [121.48, 31.22, 8675] },
              { name: "深圳", value: [114.07, 22.62, 2461] },
              { name: "广州", value: [113.23, 23.16, 187] },
              { name: "西安", value: [108.45, 34, 3421] },
            ],
          },
        ],
        tooltip: {
      }
    })
  })
}
```

```
        trigger: "item",
    },
    visualMap: {
        type: "continuous",
        min: 100,
        max: 5000,
        calculable: true,
        inRange: {
            color: ["#50a3ba", "#eac736", "#d94e5d"],
        },
        textStyle: {
            color: "#fff",
        },
    },
},
});
});
return {
    mapData,
};
},
);
},
</script>

<style lang="less">
.map {
    width: 100%;
    height: 100%;
}
</style>
```

58.图表3 产品库存统计分析图

获取数据

在components下的itemThree.vue

```
<template>
<div>
    <h2>图表1</h2>
    <div id="chartDomb" class="chart">
        </div>
    </div>
</template>

<script>
import { inject, onMounted, reactive } from "vue"
export default {
    setup() {
        // 1.设置echarts设置axios 创建接受请求的变量
        let $echarts = inject("echarts")
        let $http = inject("axios")
        let data = reactive({})
        // 2.获取数据
        async function getState() {
            let res = await $http.get("/api/statistics")
            data.list = res.data
        }
        getState()
    }
}
</script>
```

```
        data=await $http({url:"three/data"})  
    }  
  
    return{  
        getState,data  
    }  
}  
}  
</script>  
  
<style scoped>  
h2 {  
    /* 48像素 */  
    height: 0.6rem;  
    color: #ffff;  
    line-height: 0.6rem;  
    text-align: center;  
    font-size: 0.25rem;  
}  
.chart {  
    /* 高度360 */  
    height: 4.5rem;  
    /* background-color: gray; */  
}  
</style>
```

动态生成图表

```
<template>  
  <div>  
    <h2>图表1</h2>  
    <div id="chartDom" class="chart">  
    </div>  
  </div>  
</template>  
  
<script>  
import { inject, onMounted, reactive } from "vue"  
export default {  
  setup() {  
    let $echarts = inject("echarts")  
    let $http = inject("axios")  
    let data = reactive({})  
    async function getState() {  
      data = await $http({ url: "three/data" })  
    }  
    // 1. 在dom加载完毕后动态展示图表  
    onMounted(() => {  
      $echarts.init(document.getElementById("chartDom"))  
      $echarts.setOption(  
        data  
      )  
    })  
  }  
}</script>
```

```

onMounted(()=>{
    // 2.发送请求
    getState().then(()=>{
        console.log("饼状图",data.data.data)
        // 初始化echarts
        let myChart = $echarts.init(document.getElementById("chartDomb"));
        myChart.setOption({
            legend: {//设置图例
                top: 'bottom'//放到最下面
            },
            series: [
                {
                    // name: 'Nightingale Chart',
                    type: 'pie',//饼图
                    radius: [10, 100],//饼图的半径数组的第一项是内半径, 第二项是外半径
                    center: ['50%', '45%'],//饼图的中心(圆心)坐标, 数组的第一项是横坐标, 第二项是纵坐标。
                    roseType: 'area',//设置成玫瑰图
                    itemStyle: {
                        borderRadius: 10//用于指定饼图扇形区块的内外圆角半径,
                    },
                    data: data.data.data.chartData//数据
                }
            ]
        })
    })
    return{
        getState,data
    }
})
}
</script>

<style scoped>
h2 {
    /* 48像素 */
    height: 0.6rem;
    color: #fff;
    line-height: 0.6rem;
    text-align: center;
    font-size: 0.25rem;
}
.chart {
    /* 高度360 */
    height: 4.5rem;
    /* background-color: gray; */
}
</style>

```

59.类别分析图样式修改

```

<template>
<div>
    <h2>图表1</h2>
    <div id="chartDomb" class="chart">

```

```
</div>
</div>
</template>

<script>
import { inject, onMounted, reactive } from "vue"
export default {
  setup() {
    let $echarts = inject("echarts")
    let $http = inject("axios")
    let data = reactive({})
    async function getState() {
      data = await $http({ url: "three/data" })
    }
    onMounted(() => {
      getState().then(() => {
        console.log("饼状图", data.data)
        let myChart = $echarts.init(document.getElementById("chartDom"))
        myChart.setOption({
          legend: {
            top: 'bottom'
          },
          series: [
            {
              type: 'pie',
              radius: [10, 100],
              center: ['50%', '45%'],
              roseType: 'area',
              itemStyle: {
                borderRadius: 10
              },
              data: data.data.data.chartData
            }
          ],
          // 设置饼状图的颜色
          color: ['#c12e34', '#e6b600',
            '#0098d9', '#2b821d', '#005eaa', '#339ca8'],
          // 提示框，鼠标悬浮交互时的信息提示
          tooltip: {
            show: true, // 默认值 true，可选为: true (显示) | false (隐藏)
            borderRadius: 10, // 提示边框圆角，单位px，默认为4
          },
        })
      })
      return {
        getState, data
      }
    })
  }
}
</script>

<style scoped>
```

```
h2 {  
    /* 48像素 */  
    height: 0.6rem;  
    color: #fff;  
    line-height: 0.6rem;  
    text-align: center;  
    font-size: 0.25rem;  
}  
.chart {  
    /* 高度360 */  
    height: 4.5rem;  
    /* background-color: gray; */  
}  
</style>
```

60.图表4 产品月销图

数据获取

在components文件夹下的itemTwo.vue中进行设置

```
<template>  
  <div>  
    <h2>图表1</h2>  
    <div id="chartDomb" class="chart">  
      </div>  
    </div>  
  </template>  
  
<script>  
  import { inject, onMounted, reactive } from "vue"  
  export default {  
    setup() {  
      // 1. 设置echarts设置axios 创建接受请求的变量  
      let $echarts = inject("echarts")  
      let $http = inject("axios")  
      let data = reactive({})  
      // 2. 获取数据  
      async function getState() {  
        data = await $http({ url: "two/data" })  
      }  
  
      return {  
        getState, data  
      }  
    }  
  }  
</script>  
  
<style scoped>  
h2 {  
    /* 48像素 */  
}
```

```
height: 0.6rem;
color: #ffff;
line-height: 0.6rem;
text-align: center;
font-size: 0.25rem;
}
.chart {
/* 高度360 */
height: 4.5rem;
/* background-color: gray; */
}
</style>
```

动态生成基本折线图

```
<template>
<div>
<h2>图表1</h2>
<div id="chartDomc" class="chart">
</div>
</div>
</template>

<script>
import { inject, onMounted, reactive } from "vue"
export default {
  setup() {
    let $echarts= inject("echarts")
    let $http= inject("axios")
    let data=reactive({})
    async function getState(){
      data=await $http({url:"two/data"})
    }
    // 1.在dom加载完毕后动态展示图表
    onMounted(()=>{
      // 2.发送请求
      getState().then(()=>{
        console.log("折线图",data.data)
        // 初始化echarts
        let myChart = $echarts.init(document.getElementById("chartDomc"));
        myChart.setOption({
          // 设置x轴内容
          xAxis: [
            {
              type: "category",
              boundaryGap: false,//折线图与y轴距离false 没有距离
              data: data.data.data.chartData.day,
            },
            ],
          // 设置y轴内容
          yAxis: [
            {
```

```
        type: "value",
    },
],
series: [
    // 设置服饰的折线图
    {
        name: "服饰",
        type: "line",
        data:data.data.data.chartData.num.Chemicals
    },
    {
        name: "数码",
        type: "line",
        data:data.data.data.chartData.num.Clothes
    },
    {
        name: "家电",
        type: "line",
        data:data.data.data.chartData.num.Electrical
    },
    {
        name: "家居",
        type: "line",
        data:data.data.data.chartData.num.digit
    },
    {
        name: "日化",
        type: "line",
        data:data.data.data.chartData.num.gear
    },
],
})
})
return{
    getState,data
}
})
}
}
</script>

<style scoped>
h2 {
    /* 48像素 */
    height: 0.6rem;
    color: #fff;
    line-height: 0.6rem;
    text-align: center;
    font-size: 0.25rem;
}
.chart {
    /* 高度360 */
    height: 4.5rem;
    /* background-color: gray; */
}
</style>
```

61.折线图样式设置

```
<template>
<div>
    <h2>图表1</h2>
    <div id="chartDomc" class="chart">
        </div>
    </div>
</template>

<script>
import { inject, onMounted, reactive } from "vue"
export default {
    setup() {
        let $echarts = inject("echarts")
        let $http = inject("axios")
        let data = reactive({})
        async function getState() {
            data = await $http({ url: "two/data" })
        }
        onMounted(() => {
            getState().then(() => {
                console.log("折线图", data.data)
                let myChart = $echarts.init(document.getElementById("chartDomc"));
                myChart.setOption({
                    xAxis: [
                        {
                            type: "category",
                            boundaryGap: false,
                            data: data.data.chartData.day,
                        },
                    ],
                    yAxis: [
                        {
                            type: "value",
                        },
                    ],
                    series: [
                        {
                            name: "服饰",
                            type: "line",
                            data: data.data.data.chartData.num.Chemicals,
                            stack: "Total", //数据堆叠
                            smooth: true, //折线图平滑效果 变成曲线图
                            showSymbol: false, // 隐藏所有数据点
                            areaStyle: { //设置填充区域的样式
                                opacity: 0.8,
                                color: new $echarts.graphic.LinearGradient(0, 0, 0,
                                    [1, [
                                        {
                                            offset: 0,
                                            color: "rgb(128, 255, 165)",
                                        },
                                    ],
                                ],
                            }
                        }
                    ]
                })
            })
        })
    }
}
```

```
        {
          offset: 1,
          color: "rgb(1, 191, 236)",
        ],
      ],
    },
    lineStyle: { // 设置线段样式
      width: 0,
    },
    emphasis: { //设置高亮的图形样式和标签样式
      focus: "series", //只显示选中的内容高亮
    },
  },
  {
    name: "数码",
    type: "line",
    data: data.data.data.chartData.num.Clothes,
    stack: "Total", //数据堆叠
    smooth: true, //折线图平滑效果 变成曲线图
    showSymbol: false, // 隐藏所有数据点

    areaStyle: {
      opacity: 0.8,
      color: new $echarts.graphic.LinearGradient(0, 0, 0,
        [
          {
            offset: 0,
            color: "rgb(0, 221, 255)",
          },
          {
            offset: 1,
            color: "rgb(77, 119, 255)",
          },
        ],
      },
      lineStyle: { // 设置线段样式
        width: 0,
      },
      emphasis: { //设置高亮的图形样式和标签样式
        focus: "series", //只显示选中的内容高亮
      },
    },
    {
      name: "家电",
      type: "line",
      data: data.data.data.chartData.num.Electrical,
      stack: "Total", //数据堆叠
      smooth: true, //折线图平滑效果 变成曲线图
      showSymbol: false, // 隐藏所有数据点

      areaStyle: {
        opacity: 0.8,
        color: new $echarts.graphic.LinearGradient(0, 0, 0,
          [
            {
              offset: 0,
              color: "rgb(55, 162, 255)",
            },
            {
              offset: 1,
              color: "rgb(100, 149, 237)",
            },
          ],
        },
      },
    },
  ],
}
```

```
        ],
        ],
        },
        lineStyle: { // 设置线段样式
            width: 0,
        },
        emphasis: { //设置高亮的图形样式和标签样式
            focus: "series", //只显示选中的内容高亮
        },
    },
    {
        name: "家居",
        type: "line",
        data: data.data.chartData.num.digit,
        stack: "Total", //数据堆叠
        smooth: true, //折线图平滑效果 变成曲线图
        showSymbol: false, // 隐藏所有数据点
        areaStyle: {
            opacity: 0.8,
            color: new $echarts.graphic.LinearGradient(0, 0, 0,
1, [
            {
                offset: 0,
                color: "rgb(255, 0, 135)",
            },
            {
                offset: 1,
                color: "rgb(135, 0, 157)",
            },
        ],
        ],
        lineStyle: { // 设置线段样式
            width: 0,
        },
        emphasis: { //设置高亮的图形样式和标签样式
            focus: "series", //只显示选中的内容高亮
        },
    },
    {
        name: "日化",
        type: "line",
        data: data.data.chartData.num.gear,
        stack: "Total", //数据堆叠
        smooth: true, //折线图平滑效果 变成曲线图
        showSymbol: false, // 隐藏所有数据点
        areaStyle: {
            opacity: 0.8,
            color: new $echarts.graphic.LinearGradient(0, 0, 0,
1, [
            {
                offset: 0,
```

```

        color: "rgb(255, 191, 0)",
        },
        {
        offset: 1,
        color: "rgb(224, 62, 76)",
        },
        ],
        ],
        },
        linestyle: { // 设置线段样式
        width: 0,
        },
        emphasis: { //设置高亮的图形样式和标签样式
        focus: "series", //只显示选中的内容高亮
        },
        ],
        ],
        }
    )
)
return{
    getState,data
}
}
}
}
</script>

<style scoped>
h2 {
/* 48像素 */
height: 0.6rem;
color: #fff;
line-height: 0.6rem;
text-align: center;
font-size: 0.25rem;
}
.chart {
/* 高度360 */
height: 4.5rem;
/* background-color: gray; */
}
</style>

```

62.优化

设置提示框等信息

```

<template>
<div>
<h2>图表1</h2>
<div id="chartDomc" class="chart"></div>
</div>
</template>

<script>
import { inject, onMounted, reactive } from "vue";
export default {

```

```
setup() {
    let $echarts = inject("echarts");
    let $http = inject("axios");
    let data = reactive({});
    async function getState() {
        data = await $http({ url: "two/data" });
    }
}

onMounted(() => {
    getState().then(() => {
        console.log("折线图", data.data);
        let myChart = $echarts.init(document.getElementById("chartDomc"));
        myChart.setOption({
            tooltip: { //提示框组件
                trigger: "axis", //触发类型。坐标轴触发
                axisPointer: { //坐标轴指示器配置项
                    type: "cross", //指示器类型 十字准星指示器
                    label: {} //坐标轴指示器的文本标签
                },
                backgroundColor: "#e6b600", //文本标签的背景颜色就是x轴y轴上的内容
            },
            legend: {
                data: ["服饰", "数码", "家电", "家居", "日化"],
            },
            // toolbox: { //下载
            //     feature: {
            //         saveAsImage: {},
            //     },
            // },
            grid: { //组件离容器的距离
                left: "1%",
                right: "4%",
                bottom: "3%",
                containLabel: true, //grid 区域是否包含坐标轴的刻度标签
            },
            xAxis: [
                {
                    // 设置坐标轴上文字颜色
                    axisLine: {
                        linestyle: {
                            color: "#fff"
                        }
                    },
                    type: "category",
                    boundaryGap: false,
                    data: data.data.data.chartData.day,
                },
            ],
            yAxis: [
                {
                    // 设置坐标轴上文字颜色
                    axisLine: {
```

```
        lineStyle: {
          color: "#fff"
        }
      },
      type: "value",
    ],
  ],
  series: [
    {
      name: "服饰",
      type: "line",
      data: data.data.data.chartData.num.Chemicals,
      stack: "Total", //数据堆叠
      smooth: true, //折线图平滑效果 变成曲线图
      showSymbol: false, // 隐藏所有数据点

      areaStyle: {
        //设置填充区域的样式
        opacity: 0.8,
        color: new $echarts.graphic.LinearGradient(0, 0, 0, 1, [
          {
            offset: 0,
            color: "rgb(128, 255, 165)",
          },
          {
            offset: 1,
            color: "rgb(1, 191, 236)",
          },
        ]),
      },
      lineStyle: {
        // 设置线段样式
        width: 0,
      },
      emphasis: {
        //设置高亮的图形样式和标签样式
        focus: "series", //只显示选中的内容高亮
      },
    },
    {
      name: "数码",
      type: "line",
      data: data.data.data.chartData.num.Clothes,
      stack: "Total", //数据堆叠
      smooth: true, //折线图平滑效果 变成曲线图
      showSymbol: false, // 隐藏所有数据点

      areaStyle: {
        opacity: 0.8,
        color: new $echarts.graphic.LinearGradient(0, 0, 0, 1, [
          {
            offset: 0,
            color: "rgb(0, 221, 255)",
          },
          {
            offset: 1,
            color: "rgb(77, 119, 255)",
          },
        ]),
      },
    }
  ]
}
```

```
        ],
    },
    linestyle: {
        // 设置线段样式
        width: 0,
    },
    emphasis: {
        //设置高亮的图形样式和标签样式
        focus: "series", //只显示选中的内容高亮
    },
},
{
    name: "家电",
    type: "line",
    data: data.data.data.chartData.num.Electrical,
    stack: "Total", //数据堆叠
    smooth: true, //折线图平滑效果 变成曲线图
    showSymbol: false, // 隐藏所有数据点

    areastyle: {
        opacity: 0.8,
        color: new $echarts.graphic.LinearGradient(0, 0, 0, 1, [
            {
                offset: 0,
                color: "rgb(55, 162, 255)",
            },
            {
                offset: 1,
                color: "rgb(116, 21, 219)",
            },
        ]),
    },
    linestyle: {
        // 设置线段样式
        width: 0,
    },
    emphasis: {
        //设置高亮的图形样式和标签样式
        focus: "series", //只显示选中的内容高亮
    },
},
{
    name: "家居",
    type: "line",
    data: data.data.data.chartData.num.digit,
    stack: "Total", //数据堆叠
    smooth: true, //折线图平滑效果 变成曲线图
    showSymbol: false, // 隐藏所有数据点

    areastyle: {
        opacity: 0.8,
        color: new $echarts.graphic.LinearGradient(0, 0, 0, 1, [
            {
                offset: 0,
                color: "rgb(255, 0, 135)",
            },
            {
                offset: 1,
```

```
        color: "rgb(135, 0, 157)",
      ],
    ],
  },
  linestyle: {
    // 设置线段样式
    width: 0,
  },
  emphasis: {
    //设置高亮的图形样式和标签样式
    focus: "series", //只显示选中的内容高亮
  },
},
{
  name: "日化",
  type: "line",
  data: data.data.data.chartData.num.gear,
  stack: "Total", //数据堆叠
  smooth: true, //折线图平滑效果 变成曲线图
  showSymbol: false, // 隐藏所有数据点

  areaStyle: {
    opacity: 0.8,
    color: new $echarts.graphic.LinearGradient(0, 0, 0, 1, [
      {
        offset: 0,
        color: "rgb(255, 191, 0)",
      },
      {
        offset: 1,
        color: "rgb(224, 62, 76)",
      },
    ]),
  },
  linestyle: {
    // 设置线段样式
    width: 0,
  },
  emphasis: {
    //设置高亮的图形样式和标签样式
    focus: "series", //只显示选中的内容高亮
  },
},
],
),
);
return {
  getState,
  data,
};
);
},
);
};

</script>

<style scoped>
h2 {
  /* 48像素 */
}
```

```
height: 0.6rem;
color: #ffff;
line-height: 0.6rem;
text-align: center;
font-size: 0.25rem;
}
.chart {
/* 高度360 */
height: 4.5rem;
/* background-color: gray; */
}
</style>
```

63.图表5 产品库存统计图

基本柱状图

```
<template>
<div>
<h2>图表1</h2>
<div id="chartDomd" class="chart">
</div>
</div>
</template>

<script>
import { inject, onMounted, reactive } from "vue"
export default {
  setup() {
    let $echarts= inject("echarts")
    let $http= inject("axios")
    let data=reactive({})
    async function getState(){
      data=await $http({url:"four/data"})
    }
    // 1.在dom加载完毕后动态展示图表
    onMounted(()=>{
      // 2.发送请求
      getState().then(()=>{
        console.log("柱状图",data.data)
        // 初始化echarts
        let myChart = $echarts.init(document.getElementById("chartDomd"));
        myChart.setOption({
          xAxis: {
            type: "category",
            data: data.data.data.chartData.day,
          },
          yAxis: {
            type: "value",
          },
        });
      })
    })
  }
}
```

```

        series: [
          {
            name:"服饰",
            type: "bar",
            data:data.data.data.chartData.num.Chemicals,
          }
        ]
      })
    })
  return{
    getState,data
  }
}
}
}

```

</script>

```

<style scoped>
h2 {
  /* 48像素 */
  height: 0.6rem;
  color: #fff;
  line-height: 0.6rem;
  text-align: center;
  font-size: 0.25rem;
}
.chart {
  /* 高度360 */
  height: 4.5rem;
  /* background-color: gray; */
}
</style>

```

64.完成堆叠效果

```

<template>
<div>
  <h2>图表1</h2>
  <div id="chartDomd" class="chart"></div>
</div>
</template>

<script>
import { inject, onMounted, reactive } from "vue";
export default {
  setup() {
    let $echarts = inject("echarts");
    let $http = inject("axios");
    let data = reactive({});
    async function getState() {
      data = await $http({ url: "four/data" });
    }
    onMounted(() => {
      getState().then(() => {
        console.log("柱状图", data.data);
      })
    })
  }
}
</script>

```

```
let myChart = $echarts.init(document.getElementById("chartDomd"));
myChart.setOption({
    xAxis: {
        type: "category",
        data: data.data.chartData.day,
    },
    yAxis: {
        type: "value",
    },
    series: [
        {
            name: "服饰",
            type: "bar",
            data: data.data.data.chartData.num.Chemicals,
            stack: "total",//数据堆叠，同个类目轴上系列配置相同的stack值可以堆叠放置
        },
        {
            name: "数码",
            type: "bar",
            data: data.data.data.chartData.num.Clothes,
            stack: "total",//数据堆叠，同个类目轴上系列配置相同的stack值可以堆叠放置
        },
        {
            name: "家电",
            type: "bar",
            data: data.data.data.chartData.num.Electrical,
            stack: "total",//数据堆叠，同个类目轴上系列配置相同的stack值可以堆叠放置
        },
        {
            name: "家居",
            type: "bar",
            data: data.data.data.chartData.num.digit,
            stack: "total",//数据堆叠，同个类目轴上系列配置相同的stack值可以堆叠放置
        },
        {
            name: "日化",
            type: "bar",
            data: data.data.data.chartData.num.gear,
            stack: "total",//数据堆叠，同个类目轴上系列配置相同的stack值可以堆叠放置
        },
    ],
});
});
return {
    getState,
    data,
};
});
},
);
};

</script>

<style scoped>
h2 {
    /* 48像素 */
    height: 0.6rem;
    color: #ffff;
```

```
        line-height: 0.6rem;
        text-align: center;
        font-size: 0.25rem;
    }
.chart {
    /* 高度360 */
    height: 4.5rem;
    /* background-color: gray; */
}
</style>
```

65.样式优化

```
<template>
<div>
    <h2>图表1</h2>
    <div id="chartDomd" class="chart"></div>
</div>
</template>

<script>
import { inject, onMounted, reactive } from "vue";
export default {
    setup() {
        let $echarts = inject("echarts");
        let $http = inject("axios");
        let data = reactive({});
        async function getState() {
            data = await $http({ url: "four/data" });
        }
        onMounted(() => {
            getState().then(() => {
                console.log("柱状图", data.data);
                let myChart = $echarts.init(document.getElementById("chartDomd"));
                myChart.setOption({
                    xAxis: {
                        // 设置坐标轴上文字颜色
                        axisLine:{
                            linestyle:{
                                color:"#fff"
                            }
                        },
                        type: "category",
                        data: data.data.data.chartData.day,
                    },
                    yAxis: {
                        // 设置坐标轴上文字颜色
                        axisLine:{
                            linestyle:{
                                color:"#fff"
                            }
                        },
                        type: "value",
                    },
                    tooltip: {
                        trigger: "axis",
                        axisPointer: {//设置鼠标选中样式为阴影

```

```
        type: "shadow",
    },
},
legend: {},//图例
grid: {//位置
    left: "3%",
    right: "4%",
    bottom: "3%",
    containLabel: true,//设置包含坐标轴
},
series: [
{
    name: "服饰",
    type: "bar",
    data: data.data.data.chartData.num.Chemicals,
    stack: "total",
    label: {
        //图形上的文本标签，可用于说明图形的一些数据信息，比如值，名称等。
        show: true,
    },
    emphasis: {
        //高亮的图形样式和标签样式。
        focus: "series", //聚焦当前高亮的数据所在的系列的所有图形。
    },
},
{
    name: "数码",
    type: "bar",
    data: data.data.data.chartData.num.Clothes,
    stack: "total",
    label: {
        //图形上的文本标签，可用于说明图形的一些数据信息，比如值，名称等。
        show: true,
    },
    emphasis: {
        //高亮的图形样式和标签样式。
        focus: "series", //聚焦当前高亮的数据所在的系列的所有图形。
    },
},
{
    name: "家电",
    type: "bar",
    data: data.data.data.chartData.num.Electrical,
    stack: "total",
    label: {
        //图形上的文本标签，可用于说明图形的一些数据信息，比如值，名称等。
        show: true,
    },
    emphasis: {
        //高亮的图形样式和标签样式。
        focus: "series", //聚焦当前高亮的数据所在的系列的所有图形。
    },
},
{
    name: "家居",
    type: "bar",
    data: data.data.data.chartData.num.digit,
    stack: "total",
}
```

```
label: {
    //图形上的文本标签，可用于说明图形的一些数据信息，比如值，名称等。
    show: true,
},
emphasis: {
    //高亮的图形样式和标签样式。
    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: "bar",
    data: data.data.data.chartData.num.gear,
    stack: "total",
    label: {
        //图形上的文本标签，可用于说明图形的一些数据信息，比如值，名称等。
        show: true,
    },
    emphasis: {
        //高亮的图形样式和标签样式。
        focus: "series", //聚焦当前高亮的数据所在的系列的所有图形。
    },
},
],
});
});
return {
    getState,
    data,
};
);
},
},
);
},
</script>

<style scoped>
h2 {
    /* 48像素 */
    height: 0.6rem;
    color: #fff;
    line-height: 0.6rem;
    text-align: center;
    font-size: 0.25rem;
}
.chart {
    /* 高度360 */
    height: 4.5rem;
    /* background-color: gray; */
}
</style>
```

##

66.项目打包

vue项目中大家在运行的时候我们是需要用内置的devServer帮助我们自动项目 开发过程中没有问题

但是 我们所写的项目今后是需要上公网让用户访问的 所以我们需要把项目放在性能更好的服务器上运行
还有就是 我们所写的是.vue文件 浏览器不认识 没有办法直接解析

所以我们就绪要对当前项目 进行打包 就是把项目编译成 html css js 方便我们把项目放到服务器上也方便浏览器解析

打包流程

1.npm run build命令打包 但是会发现打包之后资源路径有问题

2.修改静态资源路径 publicPath

```
module.exports = {
  publicPath: process.env.NODE_ENV === 'production'
    ? './'
    : '/'
}
```

3.修改路由模式为hash

```
// 1.需要引入
import { createRouter, createWebHistory, createWebHashHistory } from 'vue-router'

// 2.修改配置
const router = createRouter({
  history: createWebHashHistory(process.env.BASE_URL),
  routes
})
```

67.服务器购买与连接

在购买ECS服务器后，系统会创建一个ECS实例。每一个ECS实例对应一台已购买的云服务器。您可以通过电脑上自带的终端工具访问云服务器，进行应用部署和环境搭建。

1. 在ECS实例列表页面，选择实例的所属地域。
2. 找到目标实例，然后在操作列选择【更多】>【密码/密钥】>【重置实例密码】，然后在弹出的对话框设置ECS实例的登录密码

实例列表

检查到安全组中包含允许对特定端口进行不受限制访问的规则，存在潜在高危风险。查看详情

实例ID/名称	标签	监控	可用区	IP地址	状态	网络类型	配置	付费方式	操作
i-uf...vm1			华东2 可用区E	10.10.9.10 (私有)	运行中	专有网络	2 vCPU 8 GiB (I/O优化) ecs.g6.large	包年包月 2020年6月15日 23:59 到期	管理 远程连接 升降配 续费 更多

启动 停止 重启 重置实例密码 线上付费转包年包月 按量付费释放设置 更多 [重置实例密码](#)

购买相同配置 实例状态 实例设置 密码/密钥 资源配比 磁盘和镜像 网络和安全组 运维和诊断 部署与弹性

3. 在弹出的页面，单击【立即重启】使新密码生效。
4. 在ECS实例列表页面，复制ECS实例的公网IP地址。
5. 连接远程桌面

(1) 方式1 浏览器直接访问

云服务器 ECS / 实例

实例

创建实例 选择实例属性项搜索，或者输入关键字识别搜索 高级搜索

实例ID/名称	标签	监控	可用区	IP地址	状态	网络类型
i-bp13sm5by8pzymwumges xixi			杭州 可用区B	118.178.233.180 (公) 172.27.170.185 (私有)	运行中	专有网络

启动 停止 重启 重置实例密码 线上付费转包年包月 按量付费释放设置 更多 共有1条

云服务器 ECS / 实例 / 实例详情

← xixi

实例详情 监控 安全组 云盘 实例快照 快照 弹性网卡 远程命令/文件 操作记录

基本信息

xixi 运行中

实例ID	i-bp13sm5by8pzymwumges
资源组	-
公网IP	118.178.233.180
安全组	sg-bp13achw... (私有)
标签	-
描述	-
地域	华东1 (杭州)
所在可用区	杭州 可用区B
主机名	xixi

转换为弹性公网IP 加入安全组 编辑标签 修改实例描述 修改实例主机名

远程连接

云服务器 ECS / 实例

概览 事件 标签 自助问题排查 发送命令/文件 (云助手) 实例 镜像 弹性容器实例 ECI 专有宿主机 DDH 云盒 超级计算集群 预留实例券 资源保障 NEW

远程连接与命令

X



经阿里云测试发现，Windows实例如果使用存在安全漏洞的老版本VirtIO驱动，在多磁盘环境下使用磁盘管理软件（ServerManager）对磁盘进行格式化操作时，可能导致非目标磁盘的数据被格式化，造成数据丢失。
为了避免实例上的数据丢失风险，建议您在阿里云云安全中心查看是否有受影响的实例，并自动修复漏洞
。您也可以使用手动修复方案>。

Workbench远程连接

通过网页可以对ECS实例进行远程控制，支持复制粘贴文本，支持多操作系统用户登录同一台实例。

立即登录

VNC远程连接

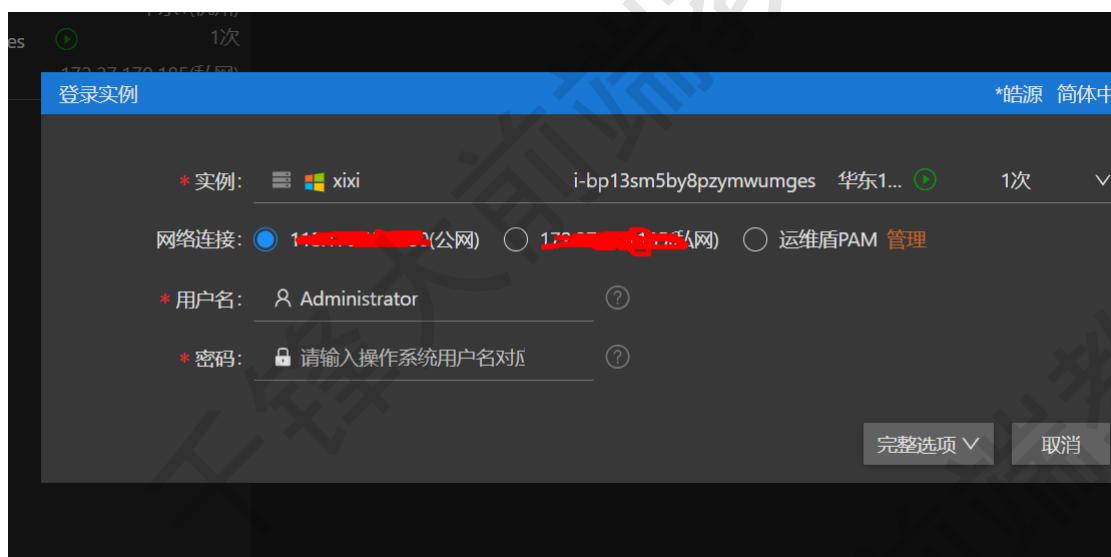
立即登录

发送远程命令 (云助手)

推荐

发送远程命令可以帮助您在实例内部快速执行命令，即无需远程连接登录实例，即可完成查看硬盘空间、安装软件、启动停止服务等操作。该功能通过[云助手](#)的命令执行功能实现，如果您的实例还没有安装或激活云助手客户端，请先[安装或者激活](#)。

发送远程命令



即可连接

(2) 远程桌面方式

在电脑的开始中搜索远程桌面



68.nginx服务器使用

Nginx是一个[http服务器](#)。是一个高性能的http服务器及反向代理服务器。官方测试nginx能够支撑5万并发链接，并且cpu、内存等资源消耗却非常低，运行非常稳定。

代理服务器

代理服务器，客户机在发送请求时，不会直接发送给目的主机，而是先发送给代理服务器，代理服务接受客户机请求之后，再向主机发出，并接收目的主机返回的数据，存放在代理服务器的硬盘中，再发送给客户机。

注意

我们学习的vue的跨域是基于脚手架内置服务器的 但是我们打包之后这个服务器就不帮助我们启动服务了 所以我们之前配置的跨域设置就作废了

使用

1.解压出nginx得到如下内容

conf	2021/11/18 16:09
contrib	2021/11/18 16:09
docs	2021/11/18 16:09
html	2021/11/18 16:09
logs	2021/2/16 23:59
temp	2021/2/16 23:59
nginx.exe	2021/2/16 22:13

2.打开conf文件夹 复制一份nginx.conf文件 并且修改名字（名字随便起）这个文件就是nginx的配置文件

fastcgi.conf	2021/2/16 23:59
fastcgi_params	2021/2/16 23:59
koi-utf	2021/2/16 23:59
koi-win	2021/2/16 23:59
mime.types	2021/2/16 23:59
nginx.conf 复制一份	2021/2/16 23:59
scgi_params	2021/2/16 23:59
uwsgi_params	2021/2/16 23:59
win-utf	2021/2/16 23:59
xixi.conf 复制的内容	2021/11/18 17:09

3.打开Powershell cd到当前nginx路径下 输入 ./nginx.exe -c ./conf/你复制的文件名.conf 启动

4.打开浏览器输入localhost:80即可启动

使用小扩展

记得如果修改服务器内容了 要停止之后在重新启动

打开Powershell cd到当前nginx路径下 输入 ./nginx.exe -c ./conf/你复制的文件名.conf -s stop 停止

69. 项目运行

1.把我们打包好的dist放到根路径下

2.修改我们的.conf文件

```
server_name localhost;

#charset koi8-r;

#access_log logs/host.access.log main;

location / {
    root dist; 我们要运行的文件夹名字
    index index.html index.htm;
}

#error_page 404 /404.html;

# redirect server error pages to the static page
#
error_page 500 502 503 504 /50x.html;
location = /50x.html {
    root html;
}
```

3.配置端口

The screenshot shows the Alibaba Cloud homepage at the top, featuring a search bar, navigation links like '合作伙伴', '支持与服务', '开发者', '了解阿里云', and '控制台' (highlighted with a red box). Below the homepage is a banner for '丰富、安全、稳定的产品及服务', with a link to '查看全部云产品'.

The main content area displays the '我的导航' (My Navigation) sidebar, which includes sections for '最近访问' (Recent Visits), '保有资源的云产品' (Cloud Products with Existing Resources), and '快捷入口' (Quick Entry). A button labeled '添加快捷入口' (Add Quick Entry) is present.

The '全部产品与服务' (All Products and Services) section features a search bar and a '产品与服务列表' (Product and Service List) table. The table shows resource statistics: 1 Cloud Server (ECS) running, 1 instance created recently, 0 expiring soon, 0 expired, and 1快照 (Snapshot). A red box highlights the '创建实例' (Create Instance) button, and a red arrow points to it with the label 'dian' (click).

The bottom section, '我的教程' (My Tutorials), lists three items: '快速搭建网站', '部署开发环境', and '搭建云上博客'.



4. 在电脑浏览器尝试使用你的公网ip加端口访问

如不行 重新启动（不要忘了先关闭nginx） 运行浏览器即可看见

70. 后端上线

同前端方式