**江西师范大学计算机信息工程学院学生实验报告**

**专业\_计算机科学与技术1班（师范）\_\_姓名\_\_\_张旭\_\_\_\_\_学号\_\_\_202126201034\_\_\_日期\_\_9.25\_\_\_\_\_\_**

|  |  |  |  |
| --- | --- | --- | --- |
| 课程名称 | Web设计 | 实验室名称 | X4313 |
| 实验名称 | 实验七 echarts的练习与使用 | | |
| 指导教师 | 吴水秀 | 成绩 |  |
| 1. 实验目的   初步了解echarts的使用方法，明白数据可视化的重要意义，自己联系逐渐掌握echarts的使用。 | | | |
| 1. 实验原理和内容   使用echarts创建五个数据可视化的案例。   * + - 1. 江西师范大学气象站，天气数据可视化       2. 自己设计一个仪表盘       3. 自己设计一个成绩雷达图       4. 自己设计一个饼状图       5. 节点分支（自己特色的）       6. 中国地图 （自己特色的） | | | |
| 1. 实验步骤   详情请见第四栏，结合代码更好讲解。  这里展示实现效果  **1.江西师大气象站**  **2. 仪表盘**  **3. 自定义雷达图** | | | |
| 4. 自定义饼状图      5. 节点分支图    6. 中国地图     1. 程序及运行结果截图（或实验数据记录及分析）    * + 1. 江西师范大学气象站   这一题是上机实验的练习，当时为了挑战自己，选择了一个看似相对较难的一个案例，结果由于css样式的问题，一直无法实现翻滚，通过查阅网上资料，终于解决了。  .wrap 类设置了 position: fixed; 和 height: 100%;。这意味着 .wrap 元素的高度被限制为视口的高度，并且它不会随着页面滚动而移动。如果 .wrap 元素包含了你的整个页面内容，这就是为什么页面不能滚动的原因。  进行修改：  .wrap{      position: relative; /\* 修改这里 \*/      left:0;      top:0;      width:100%;      height:auto; /\* 修改这里 \*/      background:#fff;  }  其实这个折线图的案例也很简单  我们只需要再原本的vue代码上添加上一些我们需要的data，以及method即可，我们再method添加上initEcharts()方法，用来初始化我们的数据图，在data中添加上date[],high[],low[]这些数据，以便后面需要，并且在queryweather()获取api接口的数据时，就进行这些数据的获得。  我是用的是map映射方法，来获取一个复杂数组中的所需要的数据。   this.date = res.data.data.map(function (forecastList) {                return forecastList.date;              });//表示日期              this.high = res.data.data.map(function (forecastList) {                return forecastList.tem\_day;              });//表示最高温度              this.low = res.data.data.map(function (forecastList) {                return forecastList.tem\_night;              });//表示最低温度              this.initEcharts();  initecharts()代码如下：   initEcharts() {          var myChart = echarts.init(document.getElementById('weather\_img'));          option = {            title: {              // text: 'Temperature Change in the Coming Week'            },            tooltip: {              trigger: 'axis'            },            legend: {},            toolbox: {              show: true,              feature: {                dataZoom: {                  yAxisIndex: 'none'                },                dataView: { readOnly: false },                magicType: { type: ['line', 'bar'] },                restore: {},                saveAsImage: {}              }            },            xAxis: {              type: 'category',              boundaryGap: false,              // data: ['Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun']              data: this.date            },            yAxis: {              type: 'value',              axisLabel: {                formatter: '{value} °C'              }            },            series: [              {                name: 'Highest',                type: 'line',                data: this.high,                markPoint: {                  data: [                    { type: 'max', name: 'Max' },                    { type: 'min', name: 'Min' }                  ]                },                markLine: {                  data: [{ type: 'average', name: 'Avg' }]                }              },              {                name: 'Lowest',                type: 'line',                // data: [1, -2, 2, 5, 3, 2, 0],                data: this.low,                markPoint: {                  data: [                    { type: 'max', name: 'Max' },                    { type: 'min', name: 'Min' }                  ]                },                markLine: {                  data: [                    { type: 'average', name: 'Avg' },                    [                      {                        symbol: 'none',                        x: '90%',                        yAxis: 'max'                      },                      {                        symbol: 'circle',                        label: {                          position: 'start',                          formatter: 'Max'                        },                        type: 'max',                        name: '最高点'                      }                    ]                  ]                }              }            ]          };          myChart.resize({            // width: '600px',            // height: '400px'          });          myChart.setOption(option);        }  这个案例主要是六个比较特殊的属性需要设置  分别是最高气温的high，最低气温的low，以及日期data，此外本案例还专门设置了一个markpoint属性，用以标记最高气温的最高点，最低气温的最低点，使用的如下代码来实现获取max与min值  data: [                    { type: 'max', name: 'Max' },                    { type: 'min', name: 'Min' }        ]  同理，还使用了同样的方法取得了数据的平均值   { type: 'average', name: 'Avg' },   * + - 1. 自定义仪表盘   感觉echarts的仪表盘比较喜欢，于是就动手实现了一个仪表盘案例  仪表盘的option关键在于type设置为gauge（仪表盘属性），data即为我们需要指定的一个指针，期中value表示我们的值，如果设置多个value可以有多个指针。为了能够区分各个指针，我们可以给不同的指针设置不同的颜色，只需要再value上添加itemstyle属性即可：  例如：    data: [                      {                          value: 70,                          itemStyle:{                              color:'red'                          }                      },                      {                          value: 90,                          itemStyle:{                              color:'blue'                          }                      },                  ]  min，max可以设置我们的指针的取值范围。  此外，echarts官网上提供的案例有一些额外的属性：如下：  axisLine: 定义了仪表盘的轴线（即仪表盘的圆圈）。  lineStyle: 定义了轴线的样式。  width: 轴线的宽度，这里设置为 30。  color: 轴线的颜色，这里设置为一个渐变色，从 '#67e0e3' 到 '#37a2da'，再到 '#fd666d'。  pointer: 定义了仪表盘的指针。  itemStyle: 定义了指针的样式。  color: 指针的颜色，这里设置为 'auto'，表示自动根据数据值计算颜色。  axisTick: 定义了仪表盘的刻度线。  distance: 刻度线距离轴线的距离，这里设置为 -30。  length: 刻度线的长度，这里设置为 8。  lineStyle: 定义了刻度线的样式。  splitLine: 定义了仪表盘的分割线（即大刻度线）。  distance 和 length: 同 axisTick。  lineStyle: 定义了分割线的样式。  axisLabel: 定义了仪表盘的刻度标签。  color: 标签的颜色，这里设置为 'inherit'，表示继承父元素的颜色。  distance: 标签距离轴线的距离，这里设置为 40。  fontSize: 标签的字体大小，这里设置为 20。  detail: 定义了仪表盘的详情（即数据值）。  valueAnimation: 是否开启数据值的动画效果，这里设置为 true。  formatter: 数据值的格式化函数，这里设置为 '{value} km/h'，表示数据值后面会添加 ' km/h'。  color: 数据值的颜色，这里设置为 'inherit'。  var myChart = echarts.init(document.getElementById('main'));      option = {          series: [              {                  type: 'gauge',                  axisLine: {                      lineStyle: {                          width: 30,                          color: [                              [0.3, '#67e0e3'],                              [0.7, '#37a2da'],                              [1, '#fd666d']                          ]                      }                  },                  pointer: {                      itemStyle: {                          color: 'auto'                      }                  },                  axisTick: {                      distance: -30,                      length: 8,                      lineStyle: {                          color: '#fff',                          width: 2                      }                  },                  splitLine: {                      distance: -30,                      length: 30,                      lineStyle: {                          color: '#fff',                          width: 4                      }                  },                  axisLabel: {                      color: 'inherit',                      distance: 40,                      fontSize: 20                  },                  detail: {                      valueAnimation: true,                      formatter: '{value} km/h',                      color: 'inherit'                  },                  data: [                      {                          value: 70,                          itemStyle:{                              color:'red'                          }                      },                      {                          value: 90,                          itemStyle:{                              color:'blue'                          }                      },                  ],                  min:0,                  max:200              }          ]      };      myChart.setOption(option);      setInterval(function () {          myChart.setOption({              series: [                  {                      data: [                          {                              value: +(Math.random() \* 100).toFixed(2)                          }                      ]                  }              ]          });      }, 2000);  这里面的setInterval是会随着时间变化而变化。   * + - 1. 自定义雷达图   雷达图可以从多个维度对数据进行对比，实际应用中会经常用到  利用echarts提供的案例进行修改，大概有以下地方需要修改：  Indicator：表示不同的属性维度  Data：value表示数值大小，name表示名字  indicator: [            { text: '语文' },            { text: '数学' },            { text: '英语' },            { text: '物理' },            { text: '化学' },            { text: '生物' }          ],  indicator: [            { text: '语文', max: 150 },            { text: '数学', max: 150 },            { text: '英语', max: 150 },            { text: '物理', max: 110 },            { text: '化学', max: 100 },            { text: '生物', max: 90 }          ],  data: [            {              value: [100, 84, 46, -80, 200,88],              name: '学生A'            },            {              value: [60, 55, 39, -100, 150,99],              name: '学生B',              areaStyle: {                color: 'rgba(255, 228, 52, 0.6)'              }            }          ]          data: [            {              value: [120, 118, 130, 100, 99, 70],              name: '学生C',              symbol: 'rect',              symbolSize: 12,              lineStyle: {                type: 'dashed'              },              label: {                show: true,                formatter: function (params) {                  return params.value;                }              }            },            {              value: [100, 93, 50, 90, 70, 60],              name: '学生D',              areaStyle: {                color: new echarts.graphic.RadialGradient(0.1, 0.6, 1, [                  {                    color: 'rgba(255, 145, 124, 0.1)',                    offset: 0                  },                  {                    color: 'rgba(255, 145, 124, 0.9)',                    offset: 1                  }                ])              }            }          ]  color: 定义了图表的颜色主题。  title: 定义了图表的标题。  text: 标题的文本内容。  legend: 定义了图表的图例，这里为空对象，表示使用默认设置。  radar: 定义了雷达图的轴线（即雷达图的“蜘蛛网”）。  indicator: 定义了轴线的标签和最大值。  center: 定义了雷达图的中心位置。  radius: 定义了雷达图的半径。  startAngle: 定义了雷达图的起始角度。  splitNumber: 定义了雷达图的分割段数。  shape: 定义了雷达图的形状。  axisName: 定义了轴线名称的样式。  splitArea: 定义了分割区域的样式。  axisLine: 定义了轴线的样式。  splitLine: 定义了分割线的样式。  series: 定义了图表的数据系列。  type: 定义了系列的类型，这里为 'radar'，表示这是一个雷达图系列。  emphasis: 定义了系列在被鼠标悬停时的样式。  data: 定义了系列的数据。  value: 数据的值。  name: 数据的名称。  areaStyle: 定义了数据区域的样式。  radarIndex: 定义了系列使用哪个雷达图，通过索引指定。  4. 饼状图  饼状图也是很常用的一种数据可视化方式，可以清晰的翻译各种数据的占比  这个实现起来也很简单。  里面唯一要注意的就是data，来自定义自己需要的数据，如果要接入vue，同样定义即可。  tooltip: 定义了鼠标悬停时显示的提示框。  trigger: 触发类型，这里设置为 'item'，表示鼠标悬停在数据项上时显示提示框。  legend: 定义了图表的图例。  top: 图例的垂直位置，这里设置为 '5%'，表示图例距离容器顶部5%的位置。  left: 图例的水平位置，这里设置为 'center'，表示图例在容器中心。  series: 定义了图表的数据系列。  name: 系列名称，用于 tooltip 的显示。  type: 系列类型，这里设置为 'pie'，表示这是一个饼图系列。  radius: 饼图的半径，这里设置为 ['40%', '70%']，表示饼图的内半径为容器宽度的40%，外半径为容器宽度的70%。  avoidLabelOverlap: 是否启用防止标签重叠策略，这里设置为 false。  itemStyle: 定义了数据项的样式。  label: 定义了数据项的标签样式。  emphasis: 定义了数据项在被鼠标悬停时的样式。  labelLine: 定义了指向数据项的线条样式，这里设置为不显示。  data: 定义了系列的数据。每个数据项都是一个对象，包含 value 和 name 属性，分别表示数据值和数据项的名称。    var myChart = echarts.init(document.getElementById('main'));    option = {    tooltip: {      trigger: 'item'    },    legend: {      top: '5%',      left: 'center'    },    series: [      {        name: 'Access From',        type: 'pie',        radius: ['40%', '70%'],        avoidLabelOverlap: false,        itemStyle: {          borderRadius: 10,          borderColor: '#fff',          borderWidth: 2        },        label: {          show: false,          position: 'center'        },        emphasis: {          label: {            show: true,            fontSize: 40,            fontWeight: 'bold'          }        },        labelLine: {          show: false        },        data: [          { value: 1048, name: '优' },          { value: 735, name: '中等' },          { value: 580, name: '及格' },          { value: 484, name: '差' },          { value: 300, name: '极差' }        ]      }    ]  };    myChart.setOption(option);   1. 节点分支图   这个图与大多数图不一样，查阅文档了解各属性含义。  linesData: [        {          coords: [            [500, 1000],            [500, 800],          ],        },        {          coords: [            [500, 800],            [100, 800],            [100, 600],          ],        },        {          coords: [            [500, 800],            [500, 600],          ],        },        {          coords: [            [500, 800],            [900, 800],            [900, 600],          ],        },        {          coords: [            [100, 600],            [0, 300],          ],        },        {          coords: [            [100, 600],            [300, 300],          ],        },        {          coords: [            [900, 600],            [700, 300],          ],        },        {          coords: [            [900, 600],            [1000, 300],          ],        },      ],  这段代码定义了一个名为 linesData 的数组，其中每个元素都是一个对象，这些对象表示一系列的线段。  每个对象都有一个 coords 属性，这是一个二维数组，表示线段的坐标。每个坐标都是一个包含两个元素的数组，第一个元素是 x 坐标，第二个元素是 y 坐标。例如，第一个对象 { coords: [[500, 1000], [500, 800]] } 表示一条从 (500, 1000) 到 (500, 800) 的线段。    for (var j = 0; j < nodes.length; j++) {      const { x, y, nodeName, svgPath, symbolSize } = nodes[j];      var node = {        nodeName,        value: [x, y],        symbolSize: symbolSize || 50,        symbol: "path://" + svgPath,        itemStyle: {          color: "orange",        },      };      charts.nodes.push(node);    }  这段代码是遍历 nodes 数组，并将每个节点的信息添加到 charts.nodes 数组中。   option = {      // backgroundColor: "",      xAxis: {        min: 0,        max: 1000,        show: false,        type: "value",      },      yAxis: {        min: 0,        max: 1000,        show: false,        type: "value",      },      series: [        {          type: "graph",          coordinateSystem: "cartesian2d",          label: {            show: true,            position: "bottom",            color: "orange",            formatter: function (item) {              return item.data.nodeName;            },          },          data: charts.nodes,        },        {          type: "lines",          polyline: true,          coordinateSystem: "cartesian2d",          lineStyle: {            type: "dashed",            width: 2,            color: "#175064",            curveness: 0.3,          },          effect: {            show: true,            trailLength: 0.1,            symbol: "arrow",            color: "orange",            symbolSize: 8,          },          data: charts.linesData,        },      ],    };  series: 定义了图表的数据系列，这里有两个系列。  第一个系列的类型是 "graph"，表示这是一个关系图。coordinateSystem 属性设置为 "cartesian2d" 表示使用二维直角坐标系。label 属性定义了节点标签的样式和内容。data 属性是节点的数据，来自 charts.nodes。  第二个系列的类型是 "lines"，表示这是一组线段。polyline 属性设置为 true 表示可以绘制多段线。coordinateSystem 属性设置为 "cartesian2d" 表示使用二维直角坐标系。lineStyle 属性定义了线段的样式。effect 属性定义了线段的动态效果。data 属性是线段的数据，来自 charts.linesData。  5. 中国地图  利用老师提供的china.js制作数据化地图，在ppcharts里面找到了一个模板。直接使用了其中的数据。  var outname = [              "南海诸岛",              "北京",              "天津",              "上海",              "重庆",              "河北",              "河南",              "云南",              "辽宁",              "黑龙江",              "湖南",              "安徽",              "山东",              "新疆",              "江苏",              "浙江",              "江西",              "湖北",              "广西",              "甘肃",              "山西",              "内蒙古",              "陕西",              "吉林",              "福建",              "贵州",              "广东",              "青海",              "西藏",              "四川",              "宁夏",              "海南",              "台湾",              "香港",              "澳门",          ];  先定义的各省份等级别的名字。  var outvalue = [0, 524, 13, 140, 75, 13, 83, 11, 19, 15, 69, 260, 39, 4, 31, 104, 36, 1052,33, 347, 9, 157, 22, 4, 18, 5, 2398, 41, 0, 484, 404, 22, 3, 5, 225,];  再给定一些数据  创造outdata   for (var i = 0; i < outname.length; i++) {              outdata.push({                  name: outname[i],                  value: outvalue[i],              });          }  在options中的data加载进我们的数据outdata | | | |
| 1. 实验小结   小结需要以如下情况总结，至少写2个：  这个小结是把做这个作业时遇到的问题，以及如何解决问题，以及总结你做作业和学习这部分内容过程中自己感受最深，受益最大，最欣赏的内容记录下来。   * + - 1. 本来尝试试试能不能找一些api来玩一玩，最后还是放弃了，觉得能够数量掌握属性就好。       2. 从最基础的折线图开始写起，慢慢的就容易掌握了。       3. 最后尝试搞了一个中国地图，其实自己根本不懂咋实现的，就是明白在哪里导入自己的数据，就可以拿模板来用了。 | | | |
|  | | | |