山东大学 计算机科学与技术 学院

大数据分析实践 课程实验报告

学号：202300130092	姓名：王俊磊	班级： 23 数据
实验题目：电子表格实践 I		
实验学时：2	实验日期：20251101	
实验目的：Add a new vis function based on the open source spreadsheet codes : https://github.com/myliang/x-spreadsheet		
软件环境：Window，vscode，利用 x-spreadsheet 进行表格操作，利用 d3 进行可视化		

实验步骤与内容：

1)核心功能概述

本实验利用开源项目 x-spreadsheet 实现在线电子表格操作功能，通过引入 D3.js 实现表格数据的可视化展示。

主要实现功能如下：1.电子表格的创建、编辑与保存；2.单元格数据的读取与监听；3.基于表格数据动态生成可视化图表；4.页面布局优化，图表与表格同时展示，界面美观简洁。

2）代码示例与说明

```
<!-- 引入所需库 -->
<link rel="stylesheet" href="https://unpkg.com/x-data-spreadsheet@1.1.5/dist/xspreadsheet.css" />
<script src="https://unpkg.com/x-data-spreadsheet@1.1.5/dist/xspreadsheet.js"></script>
<script src="https://unpkg.com/x-data-spreadsheet@1.1.9/dist/locale/zh-cn.js"></script>
<script src="https://d3js.org/d3.v6.js"></script>
```

引入所需库

```
<style>
  body {···
  }
  #xspreadsheet {
    width: 450px;
    height: 550px;
    margin-right: 30px;
    box-shadow: 0 0 10px rgba(0,0,0,0.15);
    background-color: white;
    border-radius: 10px;
    padding: 10px;
  }
  #my_dataviz {···
  }
  .checkbox {···
  }
  label {···
  }
</style>
```

定义一些画面样式

```
<div style="margin-bottom:10px;">
  <input type="checkbox" class="checkbox" value="barchart" id="barBox">
  <label for="barBox">显示柱状图</label>
  <input type="checkbox" class="checkbox" value="linechart" id="lineBox">
  <label for="lineBox">显示折线图</label>
</div>
```

可视化按钮

```
// 初始化表格
x_spreadsheet.locale("zh-cn");
var xs = x_spreadsheet("#xspreadsheet", {
  mode: 'edit',
  showToolbar: true,
  showGrid: true,
  view: { height: () => 500, width: () => 400 },
  row: { len: 15, height: 25 },
  col: { len: 8, width: 100 },
});
```

初始化表格

```
// 初始数据
xs.on('cell-edited', update);
xs.cellText(0, 1, "计算机").cellText(0, 2, "法学").reRender();
xs.cellText(1, 0, "2017").cellText(1, 1, "23").cellText(1, 2, "15").reRender();
xs.cellText(2, 0, "2018").cellText(2, 1, "36").cellText(2, 2, "26").reRender();
xs.cellText(3, 0, "2019").cellText(3, 1, "23").cellText(3, 2, "33").reRender();
xs.cellText(4, 0, "2020").cellText(4, 1, "22").cellText(4, 2, "10").reRender();
```

初始数据

```
// 主更新函数
function update() {
  const barChecked = d3.select("#barBox").property("checked");
  const lineChecked = d3.select("#lineBox").property("checked");
  d3.selectAll('svg').remove();

  if (!barChecked && !lineChecked) return;

  // 提取数据
  const yTitle = [], xTitle = [], data = [];
  let rows = 0, cols = 0;
  for (let i = 1; i < 20; i++) {
    const cell = xs.cell(i, 0);
    if (!cell || !cell.text) { rows = i; break; }
    yTitle.push(cell.text);
    data.push([]);
  }
  for (let j = 1; j < 20; j++) {
    const cell = xs.cell(0, j);
    if (!cell || !cell.text) { cols = j; break; }
    xTitle.push(cell.text);
  }

  for (let i = 1; i < rows; i++) {
    for (let j = 1; j < cols; j++) {
      const cell = xs.cell(i, j);
      if (!cell || isNaN(+cell.text)) return;
      data[i - 1][j - 1] = +cell.text;
    }
  }

  const formattedData = yTitle.map((group, i) => {
    const obj = { group };
    xTitle.forEach((x, j) => obj[x] = data[i][j]);
    return obj;
  });

  if (barChecked) drawBarChart(formattedData, xTitle, yTitle);
  if (lineChecked) drawLineChart(formattedData, xTitle, yTitle);
}
```

更新主函数，提取数据

```
// 柱状图绘制函数
function drawBarChart(data, xTitle, yTitle) {
  const margin = { top: 40, right: 40, bottom: 50, left: 60 };
  const width = 600, height = 450;
  const svg = d3.select("#my_dataviz").append("svg")
    .attr("width", width + margin.left + margin.right)
    .attr("height", height + margin.top + margin.bottom)
    .append("g").attr("transform", `translate(${margin.left},${margin.top})`);

  const subgroups = xTitle;
  const groups = yTitle;
  const maxVal = d3.max(data, d => d3.max(subgroups, k => d[k]));

  const x = d3.scaleBand().domain(groups).range([0, width]).padding(0.2);
  const y = d3.scaleLinear().domain([0, maxVal]).range([height, 0]).nice();
  const xSubgroup = d3.scaleBand().domain(subgroups).range([0, x.bandwidth()]).padding(0.05);
  const color = d3.scaleOrdinal().domain(subgroups).range(subgroups.map((_, i) => getColor(i)));

  svg.append("g").attr("transform", `translate(0,${height})`).call(d3.axisBottom(x));
  svg.append("g").call(d3.axisLeft(y));

  svg.append("g")
    .selectAll("g")
    .data(data)
    .join("g")
    .attr("transform", d => `translate(${x(d.group)},0)`)
    .selectAll("rect")
    .data(d => subgroups.map(key => ({ key, value: d[key] })))
    .join("rect")
    .attr("x", d => xSubgroup(d.key))
    .attr("y", height)
    .attr("width", xSubgroup.bandwidth())
    .attr("height", 0)
    .attr("fill", d => color(d.key))
    .transition()
    .duration(800)
    .attr("y", d => y(d.value))
    .attr("height", d => height - y(d.value));
```

柱状图绘制函数

```
// 图例
const legend = svg.selectAll(".legend")
  .data(subgroups)
  .enter().append("g")
  .attr("transform", (d, i) => `translate(0,${i * 20})`);

legend.append("rect")
  .attr("x", width - 20)
  .attr("width", 18)
  .attr("height", 18)
  .style("fill", d => color(d));

legend.append("text")
  .attr("x", width - 25)
  .attr("y", 14)
  .attr("text-anchor", "end")
  .text(d => d);
```

柱状图图例

```
// 折线图绘制函数
function drawLineChart(data, xTitle, yTitle) {
  const margin = { top: 40, right: 40, bottom: 50, left: 60 };
  const width = 600, height = 450;
  const svg = d3.select("#my_dataviz").append("svg")
    .attr("width", width + margin.left + margin.right)
    .attr("height", height + margin.top + margin.bottom)
    .append("g").attr("transform", `translate(${margin.left},${margin.top})`);

  const x = d3.scalePoint().domain(yTitle).range([0, width]);
  const maxVal = d3.max(data, d => d3.max(xTitle, k => d[k]));
  const y = d3.scaleLinear().domain([0, maxVal]).range([height, 0]).nice();

  svg.append("g").attr("transform", `translate(0,${height})`).call(d3.axisBottom(x));
  svg.append("g").call(d3.axisLeft(y));

  const color = d3.scaleOrdinal().domain(xTitle).range(xTitle.map((_, i) => getColor(i)))

  xTitle.forEach((seriesName, i) => {
    const line = d3.line()
      .x(d => x(d.group))
      .y(d => y(d[seriesName]))
      .curve(d3.curveMonotoneX);

    svg.append("path")
      .datum(data)
      .attr("fill", "none")
      .attr("stroke", color(seriesName))
      .attr("stroke-width", 2.5)
      .attr("d", line);
```

```
// 圆点
svg.selectAll(`.dot-${i}`)
  .data(data)
  .enter()
  .append("circle")
  .attr("cx", d => x(d.group))
  .attr("cy", d => y(d[seriesName]))
  .attr("r", 5)
  .attr("fill", color(seriesName));

// 标签
svg.selectAll(`.text-${i}`)
  .data(data)
  .enter()
  .append("text")
  .attr("x", d => x(d.group))
  .attr("y", d => y(d[seriesName]) - 8)
  .attr("text-anchor", "middle")
  .style("font-size", "12px")
  .text(d => d[seriesName]);
;
```

折线图绘制

```
// 图例
const legend = svg.selectAll(".legend")
  .data(xTitle)
  .enter().append("g")
  .attr("transform", (d, i) => `translate(0,${i * 20})`);

legend.append("rect")
  .attr("x", width - 20)
  .attr("width", 18)
  .attr("height", 18)
  .style("fill", d => color(d));

legend.append("text")
  .attr("x", width - 25)
  .attr("y", 14)
  .attr("text-anchor", "end")
  .text(d => d);
```

折线图图例

结论分析与体会：

通过本次实验，成功实现了基于 x-spreadsheet 的在线电子表格与 D3.js 数据可视化联动功能。用户可在网页中编辑数据并实时生成柱状图，直观展示数据变化。实验验证了前端数据处理与可视化结合的可行性，页面布局合理、美观。通过实践，掌握了开源库集成、事件监听及动态更新方法，理解了“数据—可视化—交互”的完整流程。此次实验提升了我对前端技术和可视化设计的综合运用能力，为后续复杂数据分析与展示打下了良好基础。