

学号：202300130028	姓名：苗雨健	班级：数据23
实验题目：电子表格实践 I		
实验学时：2	实验日期：2025/10/24	
<p>实验目标：</p> <p>基于开源电子表格代码（https://github.com/myliang/x-spreadsheet）新增可视化功能，掌握电子表格与数据可视化结合的实现方法，提升前端数据可视化开发能力。</p>		
<p>实验步骤：</p> <p>（一）基础 HTML 框架搭建</p> <p>实验初期需完成核心文件的引入工作，依次导入 x-spreadsheet 的 CSS 样式文件以保障表格基础样式展示，引入 x-spreadsheet 核心 JavaScript 库实现表格核心功能，加入中文语言包确保界面语言适配，同时引入 D3.js v6 版本，为后续各类图表的绘制提供技术支撑。</p> <p>（二）用户界面布局设计</p> <p>创建复选框控制面板，设置柱状图、折线图、饼图、散点图四种可视化类型的选择项，方便用户按需勾选；同时创建可视化容器#my_dataviz，专门用于承载后续生成的各类图表，实现操作区与展示区的清晰划分。界面布局核心代码如下</p> <pre><html lang="zh-CN"> <head> <meta charset="UTF-8"> <meta name="viewport" content="width=device-width, initial-scale=1.0"> <title>电子表格实践I - x-spreadsheet与D3可视化</title> <!-- 导入需要的官方库 --> <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> </head> <body> <!-- 加入多个checkbox可以选择不同的可视化类型 --> <div id="xspreadsheet"> <div class="chart-controls"> <h3>可视化选项：</h3> <label><input type="checkbox" class="checkbox" value="barchart" />柱状图</label> <label><input type="checkbox" class="checkbox" value="linechart" />折线图</label> <label><input type="checkbox" class="checkbox" value="piechart" />饼图</label> <label><input type="checkbox" class="checkbox" value="scatterchart" />散点图</label> </div> </div> <div id="my_dataviz"></div> </body> </html></pre>		
<p>（三）样式配置优化</p> <p>为保障界面美观与交互体验，对表格、图表容器及控制面板进行样式定义：</p> <p>设置#xspreadsheet容器宽 400px、高 500px，#my_dataviz容器宽 1000px、高 900px，均清除默认内外边距；</p> <p>美化复选框控制面板，添加边框、圆角及背景色，调整标题与选项的字体大小和间距；</p> <p>设置坐标轴文本样式，增强图表可读性。</p>		

```

<div id= "my_dataviz"></div>

<style>
#xspreadsheet {
  width: 400px;
  height: 500px;
  padding: 0px;
  /* border: 1px solid rgba(0, 0, 0, 0.815); */
  margin: 0px;
}
#my_dataviz {
  width: 1000px;
  height: 900px;
  padding: 0px;
  /* border: 1px solid rgba(0, 0, 0, 0.815); */
  margin: 0px;
}
.ticktext {
  font-size: 20;
  stroke: ■ rgba(0, 0, 0, 0.794);
  stroke-width: 0.05em;
}
.chart-controls {
  margin-bottom: 20px;
  padding: 10px;
  border: 1px solid ■ #771515;
  border-radius: 5px;
  background-color: ■ #655252;
}
.chart-controls h3 {
  margin: 0 0 10px 0;
  font-size: 16px;
  color: ■ #333;
}

```

（四）表格初始化与配置

语言与模式配置：设置表格为中文语言环境，确保用户操作界面友好；配置表格为edit编辑模式，显示工具栏、网格及右键菜单，方便数据录入与编辑。

行列与样式设置：设置表格行数 15 行、列数 8 列，行高 25px、列宽 100px（索引列宽 60px，最小列宽 60px）；定义表格基础样式：白色背景、左对齐、垂直居中，字体选用 Helvetica（大小 10px，默认不加粗、不倾斜）。

颜色调色板函数：编写getColor函数，提供 20 种差异化颜色，用于后续图表中不同数据系列的区分，函数代码如下

```
function getColor(idx) {
    var palette = [
        '#5ab1ef', '#ffb980', '#d87a80', '#2ec7c9', '#b6a2de',
        '#8d98b3', '#e5cf0d', '#97b552', '#95706d', '#dc69aa',
        '#07a2a4', '#9a7fd1', '#588dd5', '#f5994e', '#c05050',
        '#59678c', '#c9ab00', '#7eb00a', '#6f5553', '#c14089'
    ];
    return palette[idx % palette.length];
}

// 定义更新函数update，在该函数中，首先确认是否需要可视化，若需要可
function update() {
    // 获取所有复选框状态
    const barCheckbox = d3.select('input[value="barchart"]');
    const lineCheckbox = d3.select('input[value="linechart"]');
```

（五）主更新函数实现（数据收集与处理）

编写update函数，核心功能包括：

状态检测：获取所有复选框的选中状态，判断是否需要绘制图表；

数据读取：从表格中读取行标题（年份）、列标题（专业）及数值数据，遍历单元格时验证数据格式（确保为有效数字）；

数据存储：将处理后的data（数值数据）、xTitle（列标题）、yTitle（行标题）存储到localStorage，为图表绘制提供数据支撑。

```
// 定义更新函数update，在该函数中，首先确认是否需要可视化，若需要可视化则收集表
function update() {
    // 获取所有复选框状态
    const barCheckbox = d3.select('input[value="barchart"]');
    const lineCheckbox = d3.select('input[value="linechart"]');
    const pieCheckbox = d3.select('input[value="piechart"]');
    const scatterCheckbox = d3.select('input[value="scatterchart"]');

    const hasAnyChart = barCheckbox.property("checked") ||
        lineCheckbox.property("checked") ||
        pieCheckbox.property("checked") ||
        scatterCheckbox.property("checked");

    if (hasAnyChart) {
        var data = [];
        var ytitle = [];
        var xtitle = [];
        var col = 0;

        for (var i = 1; i < 20; i++) {
            if (xs.cell(i, 0) === null || xs.cell(i, 0).text === undefined ||
                rows = i;
                break;
            }
            data.push([]);
            ytitle.push(xs.cell(i, 0).text);
        }

        for (var i = 1; i < 20; i++) {
            if (xs.cell(0, i) === null || xs.cell(0, i).text === undefined ||
                col = i;
                break;
            }
        }
    }
}
```

（六）数据格式转换与图表绘制调度

数据还原与格式化：从localStorage读取data、xTitle、yTitle，将字符串类型数据转换为数组格式，适配图表绘制需求；

范围计算：计算数据最大值，用于设定图表 Y 轴范围，避免比例失衡；

绘制调度：清除容器中现有图表（防止重复），根据复选框选中状态，调用对应图表的绘制函

数（如drawBarChart、drawLineChart等）。

```
// 柱状图绘制函数
function drawBarChart(container, data, xTitle, yTitle, max) {
  const margin = { top: 40, right: 30, bottom: 40, left: 50 };
  const width = 600 - margin.left - margin.right;
  const height = 500 - margin.top - margin.bottom;

  const svg = container
    .append("svg")
    .attr("width", width + margin.left + margin.right + 100)
    .attr("height", height + margin.top + margin.bottom)
    .append("g")
    .attr("transform", `translate(${margin.left}, ${margin.top})`);

  const subgroups = xTitle;
  const groups = yTitle;

  const x = d3.scaleBand().domain(groups).range([0, width]).padding([0.2]);
  svg
    .append("g")
    .attr("transform", `translate(0, ${height})`)
    .call(d3.axisBottom(x).tickSizeOuter(0));

  const y = d3.scaleLinear().domain([0, max]).range([height, 0]).nice();
```

（七）各类图表绘制函数编写

1. 柱状图绘制（drawBarChart）

坐标系搭建：创建 SVG 画布（宽 600px、高 500px），设置内边距，定义 X 轴（分类比例尺，对应年份）、Y 轴（线性比例尺，对应数值）；

分组柱状图绘制：按年份分组，每个分组内绘制不同专业的柱子，通过xSubgroup比例尺控制同组内柱子间距；

标签与图例：为每个柱子添加数值标签（位于柱子顶部），调用drawLegend函数添加图例，区分不同专业数据

2. 折线图绘制（drawLineChart）

数据转换：将数据按专业分组，生成每个专业的“年份 - 数值”对应关系；

折线与数据点：使用d3.line()生成器绘制折线，在每个数据点处添加圆圈标记（增强可读性）；

样式统一：折线颜色与柱状图保持一致，调用drawLegend函数添加图例。

3. 饼图绘制（drawPieChart）

数据汇总：计算各专业的总数值，转换为“专业 - 总数值 - 占比”格式；

饼图生成：使用d3.pie()计算扇形角度，d3.arc()定义扇形形状，绘制饼图；

标签标注：在每个扇形中心添加“专业：占比 %”标签，直观展示数据占比。

4. 散点图绘制（drawScatterChart）

坐标映射：将年份映射为 X 轴位置，数值映射为 Y 轴位置，同一年份内不同专业的散点横向偏移（避免重叠）；

散点绘制：每个数据点用圆圈表示，颜色区分专业，添加数值标签；

坐标系适配：复用柱状图的 X、Y 轴比例尺逻辑，确保数据范围一致。

（八）图例绘制与事件绑定

统一图例函数（drawLegend）：为各类图表生成统一风格的图例，通过循环创建颜色方块与对应专业标签，放置于图表右侧，确保可视化风格一致。

事件绑定：将所有复选框的change事件绑定到update函数，实现“勾选 / 取消勾选”时图表实时更新，提升交互体验

结果展示：

☐ 柱状图

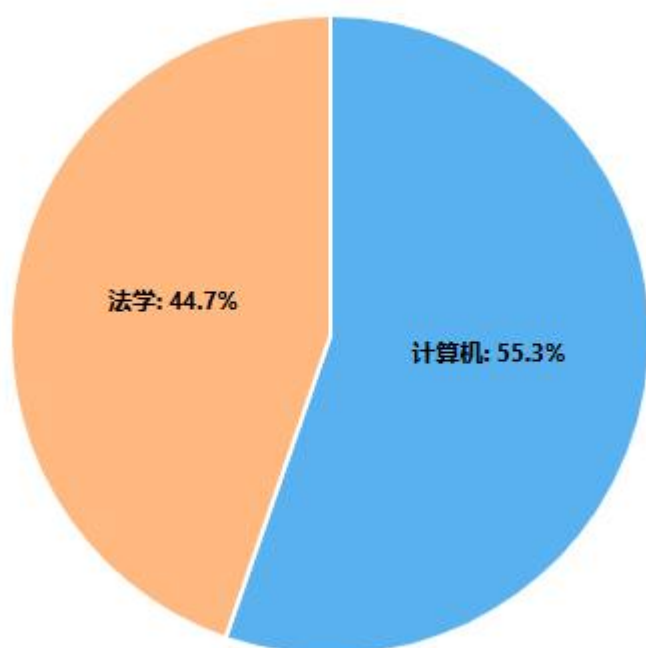
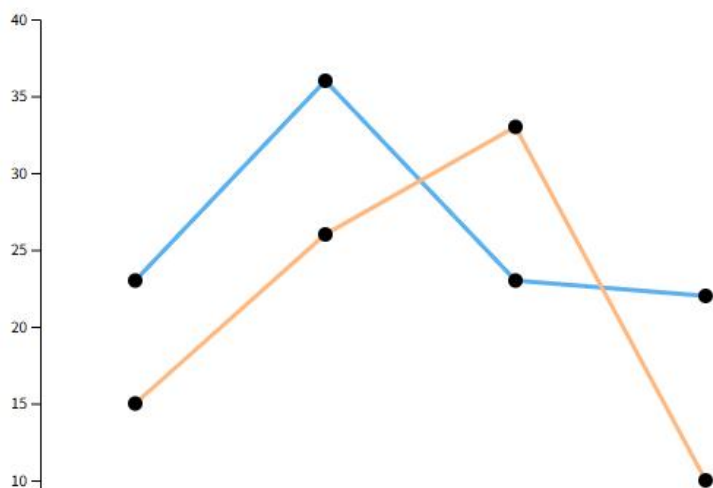
☒ 折线图

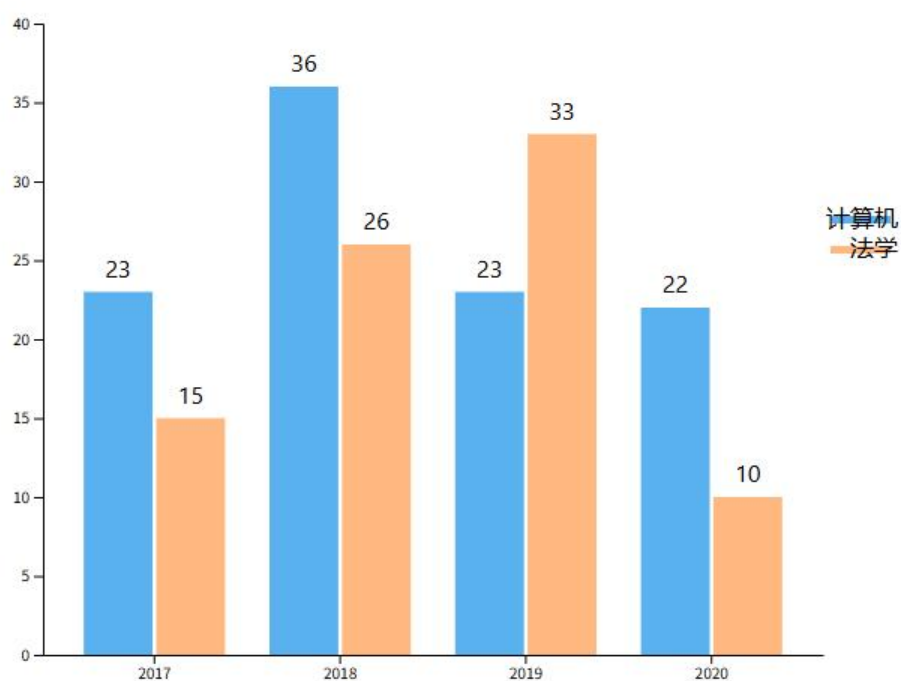
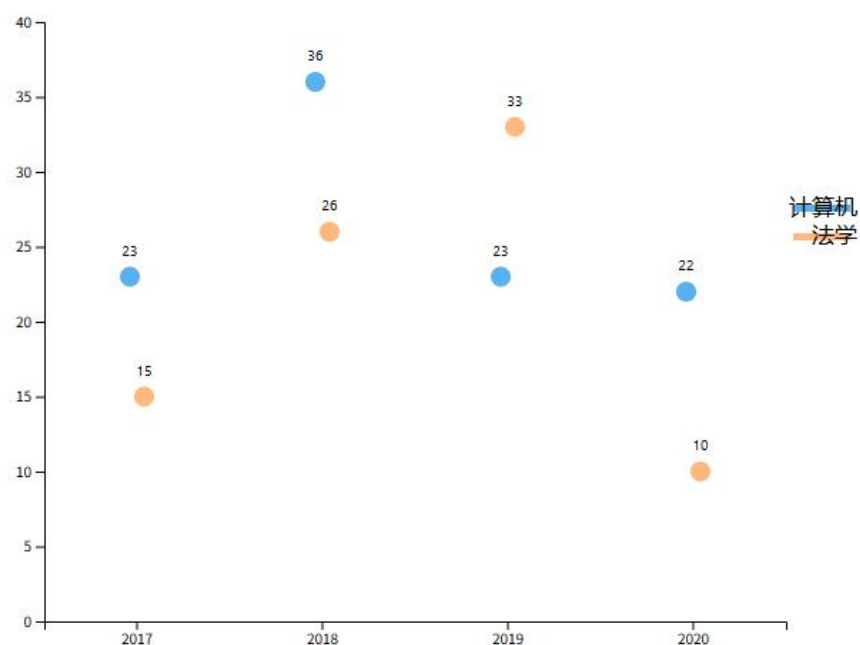
☐ 饼图

☐ 散点图

正常 Helvetica 10 B

	A	B	C	
1		计算机	法学	
2	2017	23	15	
3	2018	36	26	
4	2019	23	33	
5	2020	22	10	





结论分析与体会：

本次实验成功基于 x-spreadsheet 实现了电子表格与数据可视化的结合，完成了“数据录入 - 格式验证 - 图表绘制 - 交互更新”的全流程功能，达到了“新增可视化功能”的实验目标，所有图表均能准确反映数据特征，交互响应符合预期。

