实验报告

学号: 202300130050

姓名: 王睿

班级: 数据 23

实验题目: Libra 交互式散点图可视化

实验学时: 2

实验日期: 2025.11.15

实验目标

本次实验主要学习使用 Libra 可视化库和 D3.js 创建交互式散点图。通过结合 Libra 的交互式可视化能力和 D3.js 的数据绑定与 DOM 操作能力，实现一个功能完整的散点图可视化系统。实验旨在掌握 Libra 库的基本使用方法，理解交互式可视化的实现原理，学会将多个可视化库进行集成，并实现悬停提示和点击高亮等交互功能。

实验环境

- **操作系统**: Windows
- **开发工具**: VS Code
- **编程语言**: HTML

实验内容

1. 动态加载 Libra 库

实现 Libra 库的动态加载机制，支持多个 CDN 路径的容错处理：

- 定义多个 Libra 库的 CDN 路径（minified 和完整版本）
- 使用 ES6 模块的 `import()` 动态导入
- 实现错误处理和自动重试机制
- 通过事件系统通知库加载完成状态

关键代码逻辑：

- 尝试从第一个路径加载，失败则尝试下一个路径
- 加载成功后设置全局标志 `window.libraLoaded`
- 触发自定义事件 `libraLoadComplete` 通知其他模块

2. 生成示例数据

创建包含 120 个数据点的示例数据集（每个类别 30 个点）：

- 定义 4 个类别：A、B、C、D
- 为每个类别生成不同分布的数据点
- 每个数据点包含：x 坐标、y 坐标、类别、标签、数值、描述等属性
- 使用随机数生成器创建具有不同分布特征的数据

数据特征：

- 类别 A：x 范围约 25-175，y 范围约 25-175
- 类别 B：x 范围约 225-375，y 范围约 125-275
- 类别 C：x 范围约 425-575，y 范围约 225-375
- 类别 D：x 范围约 625-775，y 范围约 325-475

3. 实现静态可视化（坐标轴和图例）

使用 D3.js 创建散点图的基础结构：

3.1 定义比例尺

- **X 轴比例尺**：线性比例尺，根据数据范围自动计算域
- **Y 轴比例尺**：线性比例尺，根据数据范围自动计算域
- 使用 `d3.extent()` 自动计算数据的最小值和最大值

3.2 创建 SVG 容器

- 设置 SVG 尺寸：800×500 像素
- 定义边距：上 30、右 30、下 60、左 60 像素
- 创建主图形容器 `<g>`，应用平移变换

3.3 绘制坐标轴

- **X 轴**：底部坐标轴，包含轴标签"X 坐标"
- **Y 轴**：左侧坐标轴，包含轴标签"Y 坐标"
- 使用 D3 的 `axisBottom()` 和 `axisLeft()` 生成轴
- 自动生成刻度线和刻度标签

3.4 添加图例

- 在图表右下角添加图例
- 为每个类别显示颜色方块和类别标签
- 使用序数颜色比例尺定义类别颜色

4. 创建 Libra 图层并绘制数据点

4.1 绘制数据点

- 在主图形容器中绘制所有数据点
- 使用 D3 的 `selectAll().data().join()` 模式绑定数据
- 设置数据点样式：

4.2 初始化 Libra Layer

- 创建 D3Layer 类型的 Libra 图层
- 设置图层尺寸和偏移量
- 将 SVG 元素作为容器传入

关键问题解决：

- 确保数据点绘制在可见的容器中
- 处理 Libra Layer 图形容器可能为空的情况
- 实现容错机制，即使 Libra 加载失败也能显示数据点

5. 实现点击高亮交互（Libra）

使用 Libra 的交互式功能实现点击高亮：

- **继承 ClickInstrument**：使用 Libra 提供的点击交互基类
- **插入 FilterService**：根据点击的数据点类别进行过滤
- **插入 Selection Transformer**：将过滤结果转换为选择状态
- **定义高亮颜色**：使用类别颜色作为高亮色

交互流程：

1. 用户点击某个数据点

2. FilterService 根据该点的类别字段过滤数据

3. Selection Transformer 将匹配的数据点标记为选中状态

4. 选中的数据点使用高亮颜色显示

6. 实现悬停提示交互（D3.js）

使用 D3.js 的事件处理实现 Tooltip 功能：

6.1 创建 Tooltip 元素

- 创建绝对定位的 `<div>` 元素
- 设置样式：黑色半透明背景、白色文字、圆角、阴影
- 初始状态设置为不可见（opacity: 0）

6.2 绑定鼠标事件

- **mouseover**：显示 Tooltip，填充数据点详细信息
- **mousemove**：Tooltip 跟随鼠标移动
- **mouseout**：隐藏 Tooltip

6.3 Tooltip 内容

显示以下信息：

- 标签（label）
- 类别（category）
- X 坐标值（保留两位小数）
- Y 坐标值（保留两位小数）
- 数值（value）

7. 主程序入口和错误处理

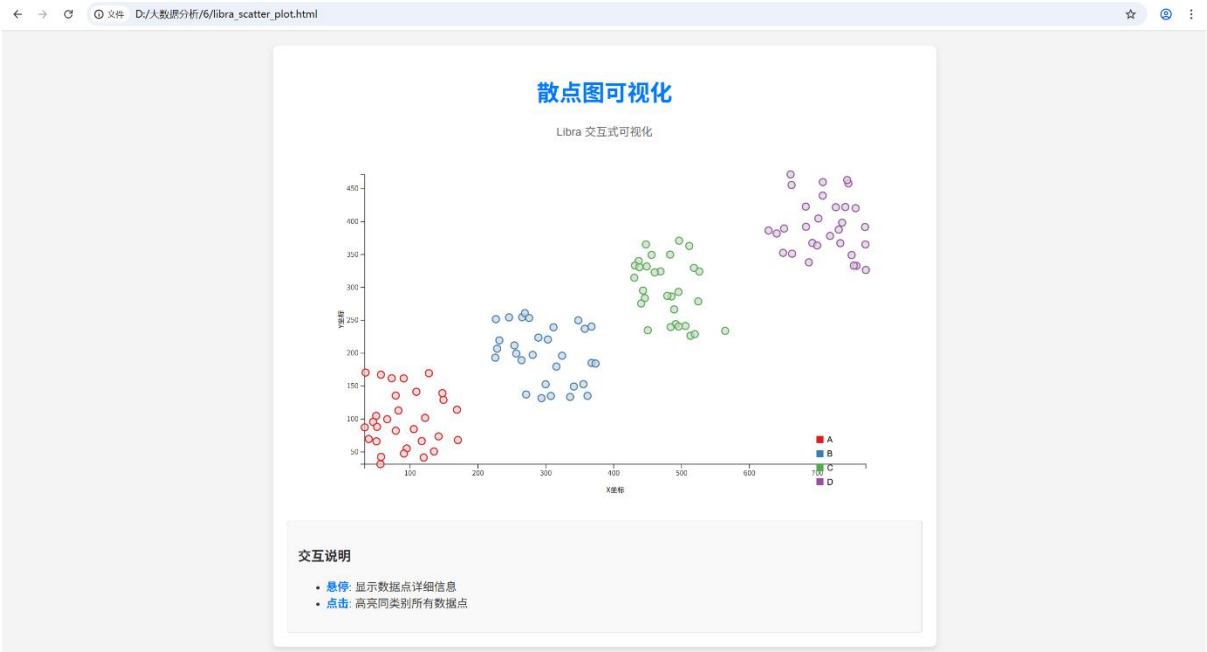
实现完整的程序流程控制：

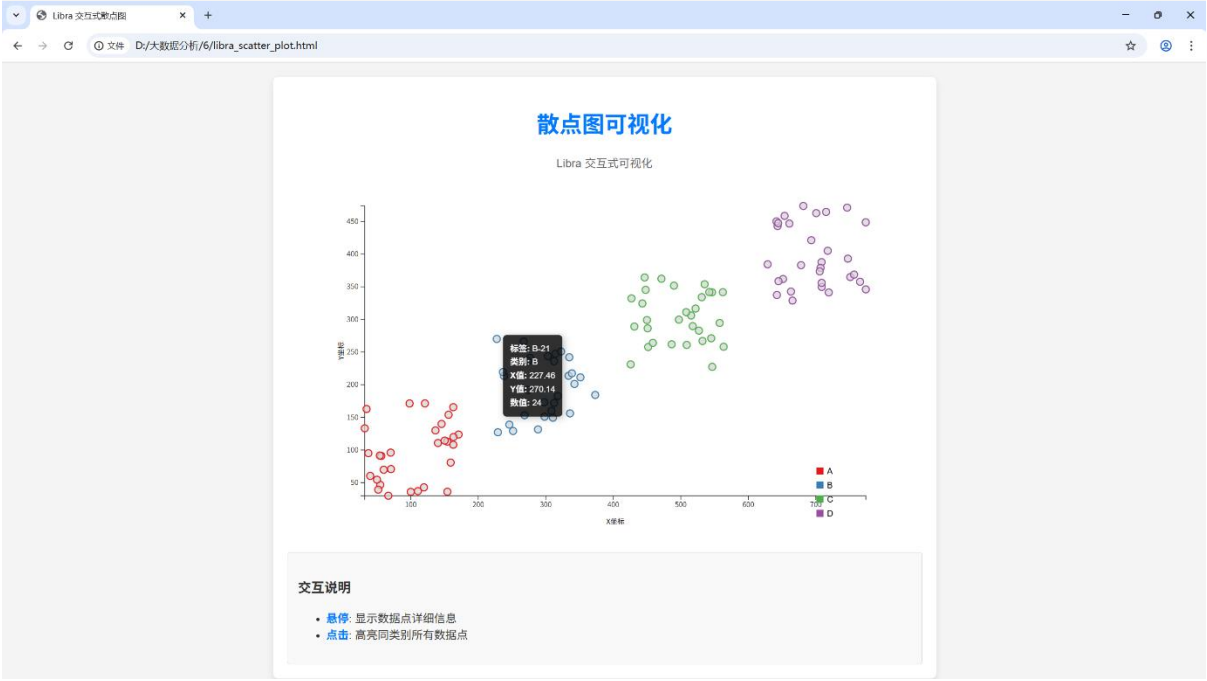
- **数据生成**：调用 `generateSampleData()` 生成示例数据
- **静态可视化**：调用 `renderStaticVisualization()` 绘制坐标轴
- **等待库加载**：使用 Promise 和事件监听等待 Libra 库加载完成
- **交互功能挂载**：库加载成功后挂载交互功能
- **错误处理**：库加载失败时使用备用方案绘制数据点

实验结果

可视化效果

成功创建了一个功能完整的交互式散点图：





技术实现亮点

1. **模块化设计**：将功能拆分为独立的函数，便于维护和调试
2. **异步处理**：正确处理 Libra 库的异步加载

3. **错误处理**：实现了完善的错误处理和容错机制
4. **库集成**：成功将 D3.js 和 Libra 两个库进行集成
5. **交互设计**：实现了两种不同类型的交互方式（悬停和点击）

遇到的问题及解决方案

问题 1：数据点不显示

问题描述：初始实现中，数据点无法显示在图表上。

原因分析：

- Libra Layer 的 `getGraphic()` 方法可能返回 `null` 或无效元素
- 数据点被绘制到了不可见的容器中

解决方案：

- 直接在 `.main-group` 容器中绘制数据点，确保可见性
- 添加容器有效性检查
- 实现备用绘制方案

问题 2：Libra 库加载失败

问题描述：某些情况下 Libra 库无法从 CDN 加载。

解决方案：

- 实现多个 CDN 路径的容错机制
- 添加库加载失败时的备用方案
- 使用事件系统通知加载状态

问题 3：Tooltip 定位问题

问题描述：Tooltip 位置不准确，可能超出屏幕范围。

解决方案：

- 使用 `event.pageX` 和 `event.pageY` 获取鼠标位置
- 添加偏移量确保 Tooltip 不遮挡数据点
- 使用 CSS 的 `pointer-events: none` 避免干扰鼠标事件

实验总结

通过本次实验，掌握了以下知识和技能：

1. **Libra 库的使用**：
2. **D3.js 高级应用**：
3. **库集成技巧**：
4. **交互式可视化设计**：
5. **问题解决能力**：

本次实验成功创建了一个功能完整、交互友好的散点图可视化系统，为后续的数据可视化项目打下了坚实的基础。通过实践，深入理解了交互式可视化的实现原理，掌握了现代 Web 可视化开发的技术栈和方法论。