第 5 章

条 形 图

学习目标

(1) 了解条形图的概念、特点和应用场景。
(2) 了解静态和动态图表的概念、特点和开发技术。
(3) 掌握利用 Web 前端开发技术制作静态条形图的方法。
(4) 掌握综合利用 Web 前后端开发技术制作动态条形图的方法。
(5) 掌握使用浏览器开发者工具进行程序调试的方法。

5.1 条形图简介

条形图是一种常用的统计图表,它用一系列长度不等的纵向或横向条纹来表现数据的分布情况。条形图的核心思想是对比,它利用人眼对高度差异敏感的特点,适合用来展示数据之间的差异。本项目拟使用条形图展示 2018 年 1 月 1 日若干监测站 $PM_{2.5}$ 浓度的日平均值,用于对比不同站点的 $PM_{2.5}$ 污染情况。为了便于读者学习,从相对简单的静态条形图入手,之后将其改造为动态版,循序渐进地介绍 ECharts 条形图的开发技术。

扫一扫

视频讲解

5.2 静态条形图

静态条形图属于纯前端的应用程序,开发技术仅涉及 HTML、CSS、JavaScript 和 ECharts。静态条形图的数据存储在前端代码中,可以直接渲染成图,而不需要经过数据抽取、转换和加载的过程,因此,开发难度相对较低,但同时功能也比较薄弱,适用于实验环境中小规模数据集的可视化。静态条形图的开发过程包括准备工作、图表制作、图表展示 3 个阶段。

5.2.1 准备工作

准备工作包括 6 个环节:①创建项目目录结构;②创建 HTML 文档;③引入 ECharts 库文件;④创建 DOM 容器;⑤设置元素样式;⑥数据准备。下面分别介绍各环节的具体

工作。

1. 创建项目目录结构

创建项目根目录 AirPollution_Bar_Static，并在根目录下分别创建 CSS 和 JS 目录，用于存放 CSS 文档和 JavaScript 脚本。

2. 创建 HTML 文档

在项目根目录下新建名为 index.html 的 HTML 文档，作为项目主页。使用 VS Code 快速生成 HTML 的基本结构，并将页面标题修改为“静态条形图”，代码如下：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>静态条形图</title>
</head>
<body>

</body>
</html>
```

3. 引入 ECharts 库文件

将 ECharts 库文件 echarts.js 放置于 AirPollution_Bar_Static/JS 目录下，并在 index.html 的<head>标签中添加引用，代码如下：

```
<script src="JS/echarts.js"></script>
```

4. 创建 DOM 容器

在 index.html 的<body>标签中定义一个 标签，将其 id 属性设置为 bar，作为条形图的容器，代码如下：

```
<div id="bar"></div>
```

5. 设置元素样式

在 AirPollution_Bar_Static/CSS 目录下新建名为 main.css 的 CSS 文档，将页面的背景颜色设置为黑色，并设置 容器的位置和尺寸，代码如下：

```
body {
    background-color: black;      /*将页面背景颜色设置为黑色*/
}

#bar {
    position: absolute;
    left: 15%;                    /*容器左边框与页面左侧的距离为页面宽度的 15%*/
    top: 15%;                     /*容器上边框与页面上部的距离为页面高度的 15%*/
    width: 70%;                   /*设置容器的宽度占页面宽度的 70%*/
    height: 70%;                  /*设置容器的高度占页面高度的 70%*/
}
```

在 index.html 的<head>标签中添加外链到 main.css 的链接，使样式生效。index.html

的完整内容如下：

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <script src="JS/echarts.js"></script>
    <link rel="stylesheet" href="CSS/main.css" />
    <title>静态条形图</title>
  </head>

  <body>
    <div id="bar"></div>
    <script src="JS/bar.js"></script>
  </body>
</html>
```

6. 数据准备

在数据准备阶段，需要根据图表的需求，采集数据并将数据整理成图表支持的格式。在本例中，需要为条形图的 x 轴组件和数据系列组件准备数据。x 轴上显示的是各监测站的编码，按照 data 属性的要求，将数据整理为如下格式：

```
["HD01","XT01","HS01","SJZ01","CZ01","BD01","LF01","TS01","QHD01","ZJK01",
"CD01",]
```

数据系列组件中存储的是各监测站 $PM_{2.5}$ 浓度的日平均值，与 x 轴上的数据一一对应，按照 data 属性的要求，将数据整理为如下格式：

```
[89.36, 149.95, 140.07, 84.82, 101.87, 71.02, 63.88, 66.33, 30.39, 21.25,16.25,]
```

5.2.2 图表制作

条形图制作阶段包括以下 5 个环节：①新建条形图 JS 脚本；②初始化条形图实例；③设置条形图配置项；④应用配置项；⑤启用自适应缩放。下面分别介绍各环节的具体工作。

1. 新建条形图 JS 脚本

在 AirPollution_Bar_Static/JS 目录下新建名为 bar.js 的 JS 脚本，用于绘制条形图，并在 index.html 的<body>标签中添加引用，代码如下：

```
<script src="JS/bar.js"></script>                    /*引用 JavaScript 脚本*/
```

2. 初始化条形图实例

在 bar.js 中，通过 document.getElementById()方法获得将被作为条形图容器的 DOM 元素，并命名为 container，然后使用 echarts.init()方法初始化一个名为 myBar 的 ECharts 实例，代码如下：

```
let container = document.getElementById('main');                //获取 DOM 容器
let myBar = echarts.init(container, null, {renderer: "svg"}); //初始化条形图实例
```

3. 设置条形图配置项

设置条形图配置项是条形图制作过程中工作量最大的环节，对于条形图最终所呈现的效果具有决定性作用。ECharts 提供了丰富的配置项，其中多数配置项都提供了默认值，完全可以胜任常规的项目需求，因此，在开发过程中，应遵循“如无必要，勿增实体”的原则，在不影响图表功能的前提下，使用尽可能少的配置项去完成开发工作，对于有默认值的配置项，能用尽用，这样做有助于控制项目的复杂度，保证项目进度。在本例中，使用了标题、提示框、工具栏、图例、网格、x 轴、y 轴、区域缩放、数据系列 9 个组件，下面分别介绍每个组件的配置方法。

1）标题组件

标题组件用于设置条形图主、副标题的内容和样式，代码如下：

```
title: {                                    //标题组件
  text: "PM2.5 浓度对比条形图",              //设置主标题文本的内容
  textStyle: {                              //设置主标题文本的样式
    color: "lightgray",                     //设置文本颜色
    fontSize: 28,                           //设置字号
  },
  subtext: "监测时间：2018 年 1 月 1 日",    //设置副标题文本的内容
  subtextStyle: {                           //设置副标题文本的样式
    color: "lightgray",
    fontSize: 18,
  },
  left: "center",                           //设置标题组件的水平位置为居中
},
```

2）提示框组件

提示框组件用于在鼠标悬停或单击数据时弹窗提示信息。为了便于观察数据，本例使用了阴影指示器，代码如下：

```
tooltip: {                                  //提示框组件
    trigger: "axis",                        //设置提示框的触发类型为坐标轴触发
    axisPointer: {                          //坐标轴指示器
        type: "shadow",                     //设置指示器类型为阴影指示器
        label: {
            show: true,                     //显示坐标轴指示器的文本标签
        },
    },
},
```

3）工具栏组件

工具栏组件提供了导出图片、重置、数据视图、数据区域缩放和动态类型切换 5 个工具，代码如下：

```
toolbox: {                                  //工具栏组件
    show: "true",                           //设置显示工具栏组件
    itemSize: 24,                           //设置工具栏图标的尺寸为 24 像素
    right: 40,                              //设置工具栏的水平位置为距离容器右侧 40 像素
```

```
        feature: {                          //各工具配置项
            saveAsImage: {show: true},      //显示“保存为图片”工具
            restore: {show: true},          //显示“配置项还原”工具
            dataView: {                     //“数据视图”工具
                readOnly: false,            //关闭只读功能
            },
            dataZoom: {show: true},         //显示“数据区域缩放”工具
            magicType: {                    //“动态类型切换”工具
                type: ["line", "bar"],      //设置图表为折线图和条形图两种动态类型切换
            }
        },
    },
```

4）图例组件

图例组件用于表达数据与图形的关联，代码如下：

```
    legend: {                           //图例组件
        show: true,                     //显示图例组件
        data: ["PM2.5 浓度"],           //设置图例对应数据系列的名称
        right: 40,                      //设置图例的水平位置
        top: 60,                        //设置图例的垂直位置
        textStyle: {                    //设置图例文本的样式
            color: "lightgray",
            fontSize: 18,
        },
    },
```

5）网格组件

条形图是平面直角坐标系下的图表，网格组件用于定义坐标系中网格的布局和样式，代码如下：

```
    grid: {                             //网格组件
        top: "20%",                     //设置网格组件的垂直位置
        left: "center",                 //设置网格组件的水平位置
        width: "80%",                   //设置网格组件的宽度
        height: "65%",                  //设置网格组件的高度
        containLabel: true,             //设置网格区域包含坐标轴的刻度标签，防止标签溢出
    },
```

6）x 轴组件

在本例中，x 轴为类目轴，用于存放各监测站的编码，代码如下：

```
    xAxis: {                            //x 轴组件
        axisLabel: {                    //设置 x 轴刻度标签
            interval: 0,                //强制显示所有标签
            color: "lightgray",         //设置刻度标签的颜色
            fontSize: 18,               //设置刻度标签的字号
        },
        data: ["HD01","XT01","HS01","SJZ01","CZ01","BD01","LF01","TS01","QHD01",
               "ZJK01","CD01",],        //数据体
    },
```

7）y 轴组件

在本例中，y 轴为数值轴，用于存放各监测站的 $PM_{2.5}$ 观测数据，代码如下：

```
yAxis: {                          //y 轴组件
    name: "单位: μg/m3",          //设置 y 轴的名称
    nameTextStyle: {              //设置 y 轴名称的样式
        color: "lightgray",
        fontSize: 18,
    },
    axisLabel: {                  //设置 y 轴刻度标签
        color: "lightgray",
        fontSize: 18
    },
},
```

8）区域缩放组件

在本例中，使用两个滑动条形区域缩放组件，分别控制 x 轴和 y 轴，代码如下：

```
dataZoom: [                       //区域缩放组件
    {
        start: 60,                //设置 x 轴的起始百分比为 60%
        end: 100                  //设置 x 轴的结束百分比为 100%
    },
    {
        yAxisIndex: 0,            //设置组件控制的 y 轴编号
        right: "5%",              //设置 y 轴滚动轴的水平位置
    },
],
```

9）数据系列组件

数据系列组件用于设置图表的类型、样式和数据体，代码如下：

```
series: [                         //数据系列组件
    {
        name: "PM2.5 浓度",       //数据系列的名称
        type: "bar",              //设置图表类型为条形图
        data: [89.36, 149.95, 140.07, 84.82, 101.87, 71.02, 63.88, 66.33, 30.39,
               21.25,16.25,],     //数据体
    },
],
```

4. 应用配置项

使用 myBar.setOption()方法将配置项应用到 myBar 实例上，并将条形图渲染到指定的 DOM 容器中，代码如下：

```
myBar.setOption(option)
```

5. 启用自适应缩放

自适应缩放是指图表能够自动改变大小去适应容器尺寸的变化，可以通过在 window.onresize 事件中调用 ECharts 实例的 resize()方法实现。window.onresize 是一个 JS 事件，用于在窗口大小发生改变时触发相应的函数。ECharts 实例的 resize()方法中提供了

animation 参数，用于启用改变大小过程中的缓动动画。实现自适应缩放的代码如下：

```
window.onresize = function () {
  myBar.resize({
    animation: {                              //改变大小时启用缓动动画
      duration: 500,                          //缓动动画的持续时间为 500 毫秒
      easing: 'linear'                        //缓动动画的效果为线性
    }
  })
}
```

最后，bar.js 的完整内容如下：

```
let container = document.getElementById("bar")
let myBar = echarts.init(container, null, {renderer: "svg"})
let option = {
  title: {
    text: "PM2.5 浓度对比条形图",
    textStyle: {
      color: "lightgray",
      fontSize: 28,
    },
    subtext: "监测时间：2018 年 1 月 1 日",
    subtextStyle: {
      color: "lightgray",
      fontSize: 18,
    },
    left: "center",
  },
  tooltip: {
    trigger: "axis",
    axisPointer: {
      type: "shadow",
      label: {
        show: true,
      },
    },
  },
  toolbox: {
    show: "true",
    itemSize: 24,
    right: 40,
    feature: {
      saveAsImage: {show: true},
      restore: {show: true},
      dataView: {
        readOnly: false,
      },
      dataZoom: {show: true},
      magicType: {
        type: ["line", "bar"],
```

```
      },
    },
  },
  legend: {
    show: true,
    data: ["PM2.5 浓度"],
    right: 40,
    top: 60,
    textStyle: {
      color: "lightgray",
      fontSize: 18,
    },
  },
  grid: {
    top: "20%",
    left: "center",
    width: "80%",
    height: "65%",
    containLabel: true,
  },
  xAxis: {
    axisLabel: {
      interval: 0,
      color: "lightgray",
      fontSize: 18,
    },
    data: ["HD01","XT01","HS01","SJZ01","CZ01","BD01","LF01","TS01",
           "QHD01","ZJK01","CD01",],
  },
  yAxis: {
    name: "单位：μg/m3",
    nameTextStyle: {
      color: "lightgray",
      fontSize: 18,
    },
    axisLabel: {
      color: "lightgray",
      fontSize: 18,
    },
  },
  dataZoom: [
    {
      start: 60,
      end: 100,
    },
    {
      yAxisIndex: 0,
      right: "5%",
    },
  ],
```

```
    series: [
      {
        name: "PM2.5 浓度",
        type: "bar",
        data: [89.36, 149.95, 140.07, 84.82, 101.87, 71.02, 63.88, 66.33, 30.39,
               21.25,16.25,],
      },
    ],
  }
  myBar.setOption(option)
  window.onresize = function () {
    myBar.resize({
      animation: {
        duration: 500,
        easing: 'linear'
      }
    })
  }
```

5.2.3 图表展示

使用浏览器打开 index.html，查看图表显示效果，如图 5.1 所示。

扫一扫

看彩图

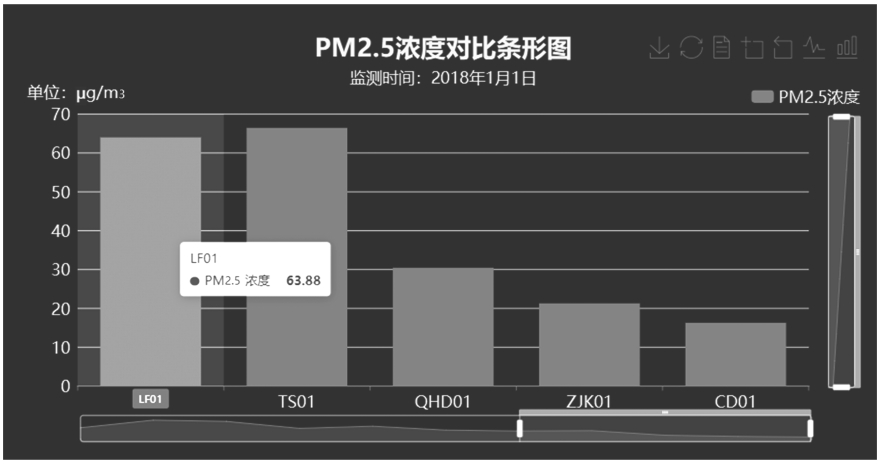

图 5.1 静态条形图

项目中各种资源的说明如表 5.1 所示。

表 5.1 项目资源列表

相对路径	说明	相对路径	说明
/CSS/main.css	CSS 文档	/JS/echarts.js	ECharts 库文件
/JS/bar.js	条形图创建脚本	/index.html	HTML 文档

扫一扫

视频讲解

5.3　动态条形图

动态条形图是一种基于 Web 服务的、重量级的应用程序，需要综合使用前后端开发技术。在本例中，动态条形图的数据源来自 MySQL 数据库，数据可视化需要经过数据抽取、转换、加载和渲染过程，因此，开发难度相对较高，但同时功能也比较强大，适用于生产环境中大规模数据集的可视化。动态条形图的开发过程同样包括准备工作、图表制作、图表展示 3 个阶段，下面分别介绍各阶段的具体工作。

5.3.1　准备工作

准备工作阶段包括 3 个环节：①创建项目目录；②配置静态资源加载路径；③配置模板文件加载路径。下面分别介绍各环节的具体工作。

1. 创建项目目录

动态条形图可以在静态版的基础上改造而成。在本地复制一份静态项目的根目录，并更名为 AirPollution_Bar_Dynamic，作为动态版的根目录。之后，在项目根目录下分别创建 static 及 templates 目录，其中 static 目录用于存放静态资源，包括 CSS 文档和 JS 脚本；templates 目录用于存放模板文件，即 HTML 文档。将 CSS 和 JS 目录移至 static 目录下，将 index.html 文档移至 templates 目录下。项目的目录结构如图 5.2 所示。

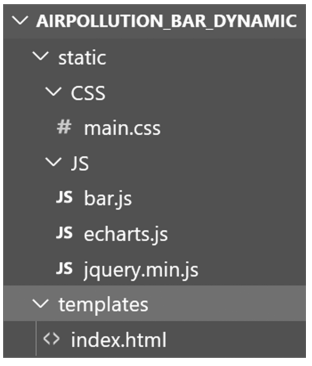

图 5.2　项目的目录结构

2. 配置静态资源加载路径

默认情况下，Flask 程序会到 static 目录下加载静态文件。在模板文件 index.html 中，使用 url_for()函数配置静态资源的加载路径，修改后的 index.html 内容如下：

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <script src="{{url_for('static',filename='JS/echarts.js')}}"></script>
    <script src="{{url_for('static',filename='JS/jquery.min.js')}}"></script>
    <script src="{{url_for('static',filename='JS/utils.js')}}"></script>
    <link rel="stylesheet" href="{{url_for('static',filename='CSS/main.css')}}">
    <title>动态条形图</title>
</head>

<body>
    <div id="bar"></div>
    <script src="{{url_for('static',filename='JS/bar.js')}}"></script>
</body>

</html>
```