余弦

一.准备数据

1. **x轴数据（类目轴）**

```
const categories=['html','css','js'];
```

2. **y轴数据**

```
 const source=[
        //html css js
        [ 30,  20, 40], //学习人数
        [ 40,  30, 50], //就业人数
]
```

3. **颜色选择**

```
const color=['#c23531','#2f4554', '#61a0a8', '#d48265', '#91c7ae','#749f83',
 '#ca8622', '#bda29a','#6e7074', '#546570', '#c4ccd3'];
```

二.建立容器

```
const main=d3.select('#main')
const width=600
const height=600
const svg=main.append('svg')
        .attr('version',1.2)
        .attr('xmlns','http://www.w3.org/2000/svg')
        .attr('width','100%')
        .attr('height','100%')
        .attr('viewBox',`0 0 ${width} ${height}`)
```

三.建立轴数据

```
/*计算类目数量 len*/
const len=categories.length

/*用range()方法，基于类目数量，获取x轴的在图表坐标系中的数据 xChartData，如[0,1,2]*/
const xChartData=d3.range(len)

/*x轴在像素坐标内的起始点和结束点 xPixelRange，左右各偏移50*/
const xPixelRange=[50,width-50]

/*-----y轴相关的基础数据-----*/
/*计算数据源中所有数据的极值 maxY
    *    用js原生方法flat()展开数据源，再通过max()方法取极值
    * */
const maxY=Math.max(...source.flat())

/*声明y轴在图表坐标系中的数据起点和结束点 yChartRange*/
```

```
const yChartRange=[0,maxY]

/*声明y轴在像素坐标系中的数据起点和结束点 yPixelRange*/
const yPixelRange=[height-50,50]
```

四.建立比例尺

1. 建立x轴比例尺

```
/*
    * 用scaleBand()方法建立分段比例尺 xScale
    * 用domain()方法在比例尺中写入图表数据xChartData
    * 用rangeRound()方法在比例尺中写入像素数据，即像素的起始位和结束位xPixelRange
    * 用padding()方法设置类目的内边距，百分比单位，如0.1
    * */
    const xScale=d3.scaleBand()
        .domain(xChartData)
        .rangeRound(xPixelRange)
```

2. 建立y轴比例尺

```
 /*-----y 轴比例尺 xScale-----*/
    /*
    * 用scaleLinear()方法建立线性比例尺 yScale
    * 用domain()方法在比例尺中写入图表数据yChartRange
    * range()方法在比例尺中写入像素数据，即像素的起始位和结束位yPixelRange
    * */
    const yScale=d3.scaleLinear()
        .domain(yChartRange)
        .range(yPixelRange)
```

5.建立轴对象

1. x轴对象

```
const xAxisGenerator=d3.axisBottom(xScale)

    /*利用坐标轴生成器绘制坐标轴
    *    在svg中append 加入g 对象
    *    用transform 属性中的translateY 设置x轴的y位置
    *    用call()方法调用xAxisGenerator轴生成器，生成坐标轴
    *    用selectAll()方法选择所有的text文本
    *    用text()方法将图表数据设置为类目数据
    *    用attr()方法设置字体大小
    * */
    svg.append('g')
        .attr('transform', `translate(0, ${height-50})`)
        .call(xAxisGenerator)
        .selectAll('text')
        .text(n=>categories[n])
        .attr('font-size','12px')
```

2. y轴对象

```
/*基于比例尺yScale，用axisLeft()方法创建刻度朝左的坐标轴生成器 yAxisGenerator*/
    const yAxisGenerator=d3.axisLeft(yScale)


    /*利用坐标轴生成器生成坐标轴
    *    在svg中append 加入g 对象
    *    用transform 属性中的translate设置y轴的x位置
    *    用call()方法调用xAxisGenerator轴生成器，生成坐标轴
    *    用attr()方法设置字体大小
    * */
    svg.append('g')
        .attr('transform','translate(50 0)')
        .call(yAxisGenerator)
        .attr('font-size','12px')
```

6.划分图标的单个柱状图宽度和颜色

```
 /*用x轴比例尺xScale的bandwidth()方法获取x轴上一个类目的像素宽xBandW*/
    const xBandW=xScale.bandwidth()
    console.log('xBandW',xBandW);
    /*获取系列的数量n*/
    const n=source.length

    /*用类目宽除以系列数，得到一个类目中每个系列元素的宽，即列宽colW*/
    const colW=xBandW/n
    console.log('colW',colW);

    /*计算调色盘颜色数量colorLen*/
    const colorLen=color.length
```

7.架构绘图区

```
/*在svg中建立系列集合seriesObjs，在系列集合中建立系列对象
    *    在svg中append 加入g 对象
    *    selectAll() 选择所有g元素，此处重点不在选择，而是建立一个选择集对象
    *    用data() 方法将具备系列信息的数据源source绑定到系列集合中
    *    用join() 基于数据源批量创建g元素，一个g代表一个系列，之后每个g元素里都会放入三个不同类目的柱状体
    *    用transform 属性中的translate设置系列的x像素位——列宽乘以系列索引
    *    基于系列索引，从调色盘中取色，然后将其作为一个系列中所有图形的填充色
    * */
    const seriesObjs=svg.append('g')
        .selectAll('g')
        .data(source)
        .join('g')
        .attr('transform',(seriesData,seriesInd)=>{
            const seriesX=colW*seriesInd
            return `translate(${seriesX},0)`
        })
        .attr('fill',(seriesData,seriesInd)=>color[seriesInd%colorLen])

    /*在系列集合中建立柱状体集合rects
    *    用系列集合seriesObjs 的selectAll()方法选择所有的rect元素，用于建立选择集对象
    *    用data()方法将之前绑定在每个系列集合中的数据绑定到柱状体集合中
```

```
    *    用join()基于每个系列的数据数据批量创建rect元素
    *    用classed() 方法为其添加item属性
    * */
    const rects=seriesObjs.selectAll('rect')
        .data(seriesData=>seriesData)
        .join('rect')
        .classed('item',true)

    console.log('rects',rects);
```

8.设置主体的宽高

```
/*=8-用attr()方法设置每个柱状体的x、y位置和width、height 尺寸=*/
    /*
    * 设置柱状体的x像素位
    *    从回调参数中获取柱状体在当前系列中的索引rectInd,系列索引 seriesInd
    *    基于柱状体在当前系列中的索引rectInd，用x轴比例尺xScale()获取柱状体在当前系列中的x像素位
    * 设置柱状体像素宽width为列宽colW
    * 设置柱状体的y像素位
    *    从回调参数中解构柱状体数据rectData
    *    基于柱状体数据rectData，用y轴比例尺yScale()获取柱状体的y像素位
    * 设置柱状体的像素像素高
    *    从回调参数中解构柱状体数据rectData
    *    让y轴上刻度为0的像素位，减去刻度为柱状图实际数据的像素位，即为柱状图的像素高
    * */
    const yKKBScale=KKBScale(0,height-50,maxY,50)
    console.log('yKKBScale',yKKBScale(2));
    console.log('yScale',yScale(2));

    rects.attr('x',(rectData,rectInd)=>xScale(rectInd))
        .attr('width',colW)
        .attr('y',rectData=>yKKBScale(rectData))
        .attr('height',rectData=>yKKBScale(0)-yKKBScale(rectData))



    /*自建比例尺*/
    function KKBScale(ax,ay,bx,by){
        const AB={x:bx-ax,y:by-ay}
        const k=AB.y/AB.x
        const b=ay-ax*k
        return function(x){
            return k*x+b
        }
    }
```