暗号：反正弦

传送门：本文是原创（非copy他人博文），写完后直接发布到掘金上，图片资源没有保存到本地，因此会有水印

echarts图表绘制

echart绘图步骤

1.建立dom 容器（**必须指定元素宽高**）

2.引入 ECharts（CDN/npm都行）

3.实例化echarts

```
echarts.init(dom);
```

4.建立图表配置项（就是图表建立工程）

- 建立坐标系
- 定义系列点
  - 数据
  - 图表类型
  - 系列图形样式设置
- 组件装配

5.显示图表

```
myChart.setOption(option);
```

```
<style>
```

```
    #main{
        width: 700px;
        height: 500px;
    }
</style>
<!--建立dom 容器-->
<div id="main"></div>
<!--引入echarts-->
<script src="https://lib.baomitu.com/echarts/4.7.0/echarts.min.js"></script>
<script>
    /*基于准备好的dom，初始化echarts实例
    *   echarts.init(dom)
    * */
    const dom=document.getElementById('main');
    const myChart=echarts.init(dom);

    /*指定图表的option 配置项和数据
    *   xAxis x轴 {}
    *       data 类目数据
    *   yAxis y轴 {}
    *   series 系列列表
    *       type 图表类型
    *       data 数据，与xAxis.data 相对应
    * */
    const option={
        xAxis:{
            data:['html','css','js']
        },
        yAxis:{},
        series:{
            type:'bar',
            data:[30,20,40]
        }
    }

    /*使用刚指定的配置项和数据显示图表
    *   setOption(option)
    * */
    myChart.setOption(option);

</script>
```

echart坐标系图表类型和组件

> echarts中所有图表（坐标系/类型/组件）都是通过定义配置项option来完成的，具体用法官网有说明，用到的时候直接CV就好，这里就不一一举例了
>
> 需要注意的是像地图，树图等需要先获得数据再进行配置，或者先进行配置画图，数据完成后再setOption下，细节部分请看高级应用的异步数据

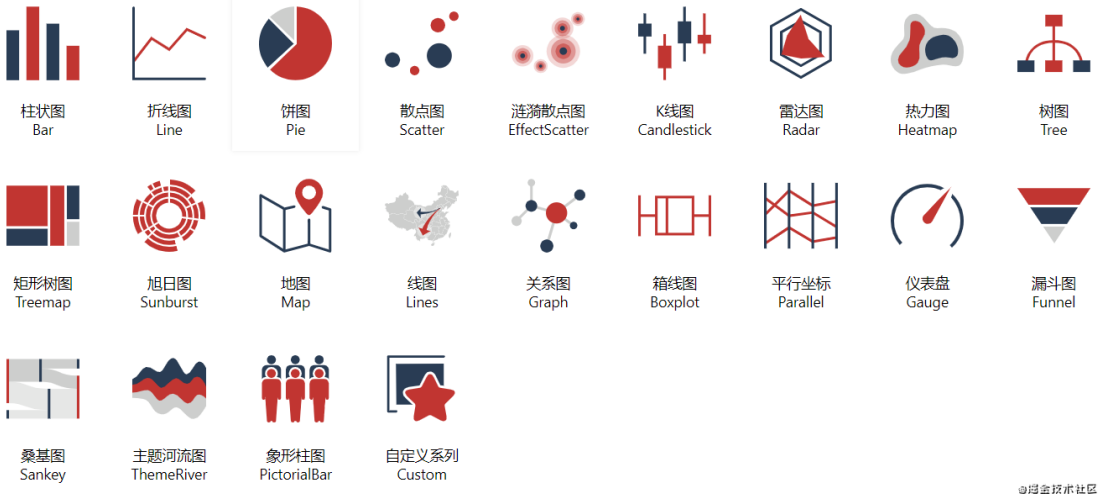
[图表和组件配置项具体用法]

[echarts图表和组件速查]

- 坐标系

- 图表类型

  下图为各图表示意图及type属性名



- 图表组件

  下图为可能用到的组件（组件在echarts中指的是option配置项），如果你经常和excel或者PPT打交道，前6个想必应该很熟悉吧：

  - 标题和图列：就不用多说和说法excel完全一致；
  - 提示框：类比excel图表就是数据标签；
  - 标注：这个excel没有，是在菜单栏插入-->形状-->标注，通常对于某一数据特别说明
  - 标线：类比excel中的数据中的标准线
  - 标语：这个excel没有，PPT里常用在菜单栏插入-->形状-->矩形（置于底层），突出数据

echarts常用图表

在处理分析数据过程中，常用图表如下

- 柱状图和折线图：直观的说明数据和相关因素的关系
- 散点图：直观反映在坐标系中位置或者分析数据的聚集状况
- 饼图：直观反映各项因子占比
- 雷达图：直观反映某个项目中，各部分实力情况
- K线图：股票用

- 树图：思维导图
- 地图：地理区域数据的可视化
  - 地图的绘制

    1.下载地图文件

    2.注册地图：echarts.registerMap('china', data);

    3.配置地图：

    series: [{

    type: 'map',

    map: 'china'

    }]
  - 【扩展】地理坐标系组件geo

    > geo 和map 的区别在于，geo支持在地理坐标系上绘制散点图，线集等

    geo用法

echarts高级应用

多坐标轴

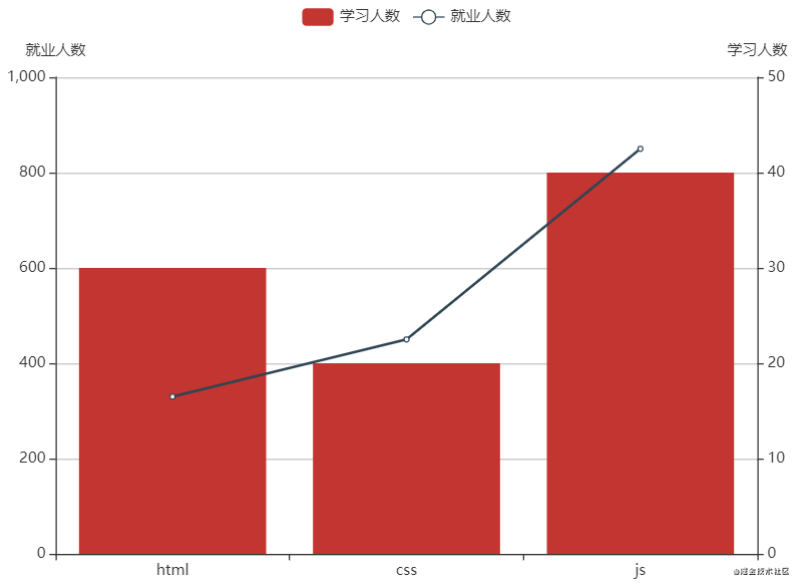
多坐标轴的常见应用就是一个图表有两个y 轴。

- 设置两个y轴，y轴的刻度线个数保持一致
- series设置两个，通过yAxisIndex 将series和yAxis对应

对于坐标轴需**重点了解yAxisIndex（series）**，下面的例子很好的说明

```
/*指定图表的配置项和数据*/
const option = {
    /*图例*/
    legend: {
        data: ['学习人数', '就业人数']
    },
    /*提示*/
    tooltip: {},
    /*x 轴*/
    xAxis: {
        data: ['html', 'css', 'js']
    },

    /*y 轴
     *   name 坐标轴名称
     *   min 刻度最小值
     *   max 刻度最大值
     * */
    yAxis: [
        {
            name: '就业人数',
            min: 0,
            max: 1000
        },
        {
```

```
            name: '学习人数',
            min: 0,
            max: 50
        },
    ],

    /*系列列表 series
     *   yAxisIndex 当前系列对应的y 轴的索引位置
     * */
    series: [{
            name: '学习人数',
            type: 'bar',
            data: [30, 20, 40],
            yAxisIndex: 1
        },
        {
            name: '就业人数',
            type: 'line',
            data: [330, 450, 850],
            yAxisIndex: 0
        }
    ]
};
```



异步数据

请求数据的方式：ajax、fetch 都可以，这是js 基础，就不再多说。

数据的更新有两种思路：

- 请求到数据后，setOption()

```
fetch('./data/China.json')
    .then((res) => res.json())
    .then(data => {
        /*注册地图*/
        echarts.registerMap('china', data);
        /*配置项*/
        const option = {
```

```
            title: {
                text: '中国地图',
                left:'center'
            },
            series: {
                type: 'map',
                map: 'china'
            }
        };
        /*基于配置项显示图表*/
        myChart.setOption(option);
    })
```

- 先setOption()，有什么先配置什么。等请求到数据后，再追加配置

```
myChart.setOption({
    title: {
        text: '中国地图',
        left:'center'
    }
});
myChart.showLoading();
fetch('./data/China.json')
    .then((res) => res.json())
    .then(data => {
        myChart.hideLoading();
        /*注册地图*/
        echarts.registerMap('china', data);
        /*等请求到数据后，追加配置*/
        myChart.setOption({
            series: {
                type:'map',
                map:'china',
            }
        });
    })
```

注：在数据加载的过程中，还可以使用echarts 实例对象的loading 功能

- 显示 loading：`myChart.showLoading()`
- 隐藏 loading：`myChart.hideLoading()`

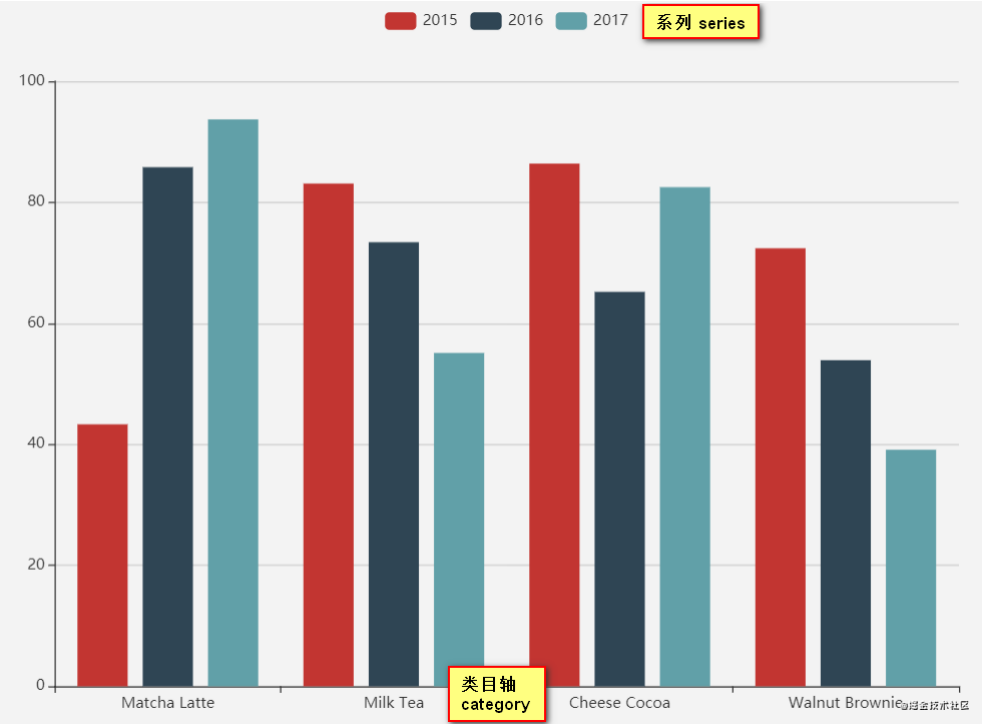
数据集

dataset 数据集组件是从ECharts 4 开始有的，用于数据管理。

dataset 的优点：

- 基于原始数据，设置映射关系，形成图表。
- 数据和配置分离，便于单独管理。
- 数据可以被多个系列或者组件复用。
- 支持更多的数据的常用格式，例如二维数组、对象数组等。

对于数据集**需要重点理解 *类目轴 category* 和 *系列 series***



数据格式说明

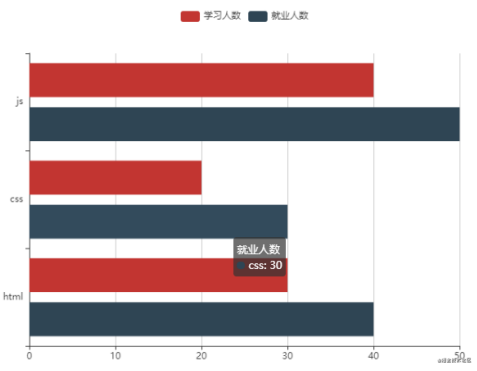
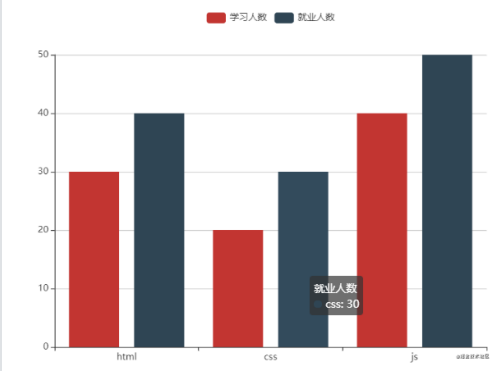
数据测试地址

```
dataset.source
//原始数据可以有以下三种形式

/*
*形式一：二维数组
*可以将数据当成表格，第一行数据是定义系列 series，第一列数据定义的是类目轴 category（默认，可通过seriesLayoutBy修改）
*/
dataset.source=[
    ['product', '2015', '2016', '2017'],//其中第一行/列可以给出维度名，第一行表示图例，可以不填；第一列表示x轴类目，必填
    ['Matcha Latte', 43.3, 85.8, 93.7],
    ['Milk Tea', 83.1, 73.4, 55.1],
    ['Cheese Cocoa', 86.4, 65.2, 82.5],
    ['Walnut Brownie', 72.4, 53.9, 39.1]
]

/*
*形式二：按行的key-value 形式（对象数组）
*数组中每个对象第一个属性 定义的是类目轴 category，之后每一个key定义 系列 series（默认，可通过seriesLayoutBy修改）
*/
dataset.source=[
    {product: 'Matcha Latte', count: 823, score: 95.8},
    {product: 'Milk Tea', count: 235, score: 81.4},
    {product: 'Cheese Cocoa', count: 1042, score: 91.2},
    {product: 'Walnut Brownie', count: 988, score: 76.9}
]
/*
*形式三：按列的 key-value 形式，每一项表示二维表的 “一列”：
*对象第一个属性 定义的是类目轴 category ，之后每一个key定义 系列 series（默认，可通过seriesLayoutBy修改）
```

```
*/
dataset.source={
    'product': ['Matcha Latte', 'Milk Tea', 'Cheese Cocoa', 'Walnut Brownie'],
    'count': [823, 235, 1042, 988],
    'score': [95.8, 81.4, 91.2, 76.9]
}
```

数据集简单案例

```
//数据源
const source=[
    ['大前端','学习人数','就业人数'],
    ['html',  30,       40],
    ['css',  20,       30],
    ['js',  40,       50],
]
// 指定图表的配置项和数据
const option = {
    tooltip:{},
    legend:{data:['学习人数','就业人数']},
    /*
    * dataset数据集
    *   source 数据源 []
    * */
    dataset:{source},

    /*x轴
    *   type 轴的类型
    *       category 类目轴，离散型数据
    *       value 数值轴，连续性数据
    * */
    xAxis:{
        type:'category'
    },
    //xAxis:{
    //    type: 'value'
    //},

    /*系列列表*/
    yAxis:{
        type: 'value'
    },
    //yAxis:{
    //    type:'category'
    //},

    series:[
        {type:'bar'},
        {type:'bar'},
    ]
};
```

轴类型	y轴是类目轴	x轴是类目轴
xAxis	type: 'value'	type:'category'
yAxis	type:'category'	type: 'value'
图表		

> 我们制作数据可视化图表的逻辑是这样的：基于数据，在配置项中指定如何映射到图形。
> 概略而言，可以进行这些映射：
>
> - 指定 dataset 的列（column）还是行（row）映射为图形系列（series）。这件事可以使用 `series.seriesLayoutBy` 属性来配置。默认是按照列（column）来映射。
> - 指定维度映射的规则：如何从 dataset 的维度（一个“维度”的意思是一行/列）映射到坐标轴（如 X、Y 轴）、提示框（tooltip）、标签（label）、图形元素大小颜色等（visualMap）。这件事可以使用 `series.encode` 属性，以及 `visualMap` 组件（如果有需要映射颜色大小等视觉维度的话）来配置。上面的例子中，没有给出这种映射配置，那么 ECharts 就按最常见的理解进行默认映射：X 坐标轴声明为类目轴，默认情况下会自动对应到 dataset.source 中的第一列；三个柱图系列，一一对应到 dataset.source 中后面每一列。

行列映射

[数据测试地址]

行列映射就是把数据集（ dataset ）的行或列映射为系列（series），配置步骤如下

1.定义两个坐标系

```
//gridIndex : x/y 轴所在的 grid 的索引，默认位于第一个 grid，值为0。
grid: [
    {bottom: '55%'},
    {top: '55%'}
],
xAxis: [
    {type: 'category', gridIndex: 0},
    {type: 'category', gridIndex: 1}
],
yAxis: [
    {gridIndex: 0},
    {gridIndex: 1}
],
```

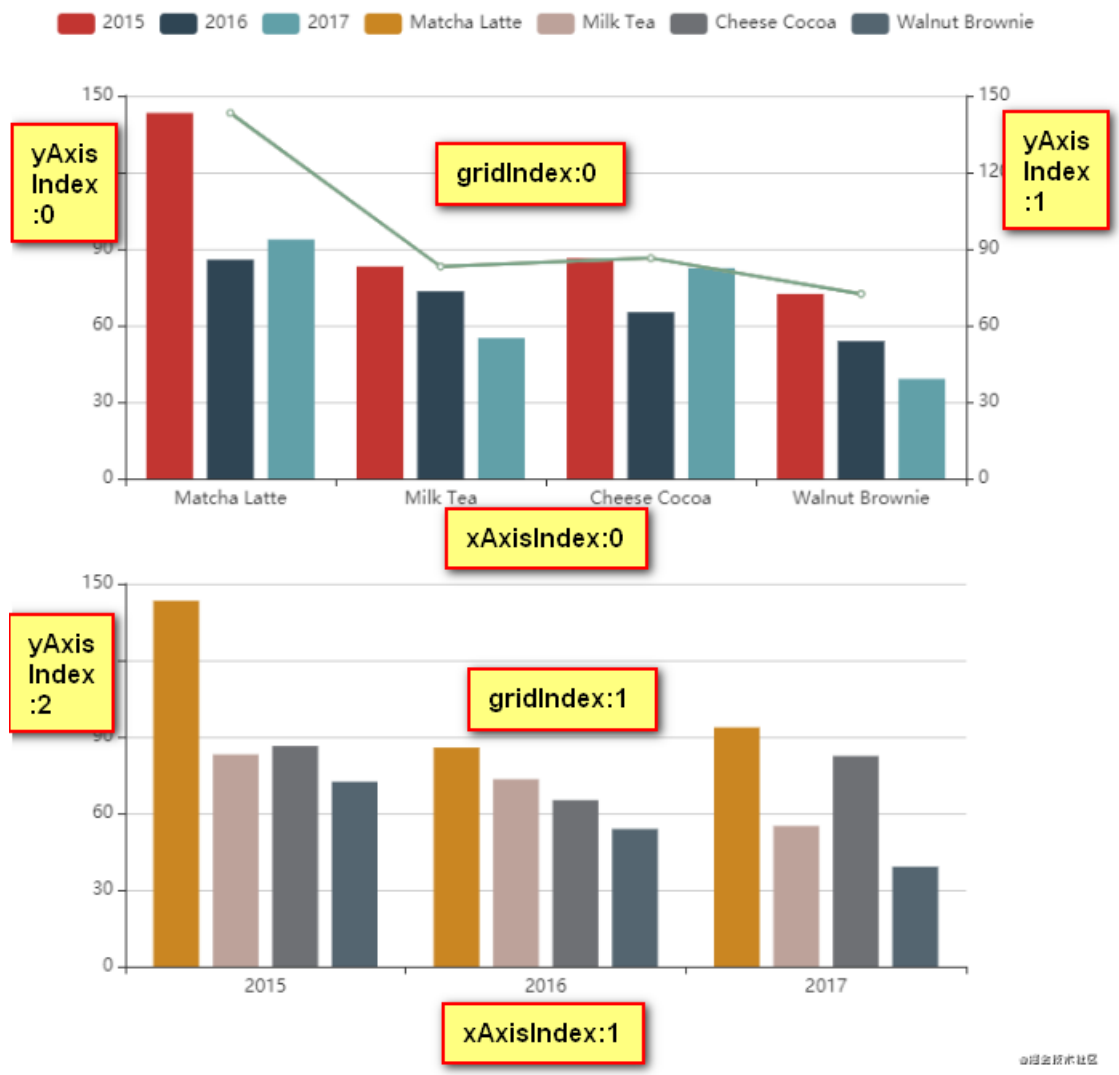
2.定义系列的数据

```
series: [
    // These series are in the first grid.
    {type: 'bar', seriesLayoutBy: 'row'},
    {type: 'bar', seriesLayoutBy: 'row'},
    {type: 'bar', seriesLayoutBy: 'row'},
    // These series are in the second grid.
    {type: 'bar', xAxisIndex: 1, yAxisIndex: 1},
    {type: 'bar', xAxisIndex: 1, yAxisIndex: 1},
    {type: 'bar', xAxisIndex: 1, yAxisIndex: 1},
    {type: 'bar', xAxisIndex: 1, yAxisIndex: 1},
]
```

案例

```
const source = [
    ['product', '2015', '2016', '2017'],
    ['Matcha Latte', 143.3, 85.8, 93.7],
    ['Milk Tea', 83.1, 73.4, 55.1],
    ['Cheese Cocoa', 86.4, 65.2, 82.5],
    ['Walnut Brownie', 72.4, 53.9, 39.1]
];
const option = {
    legend:{},
    dataset:{source},
    xAxis:[
        {type:'category',gridIndex:0},
        {type:'value',gridIndex:1},
    ],
    yAxis:[
        {type:'value',gridIndex:0},
        {type:'value',gridIndex:0},
        {type:'category',gridIndex:1},
    ],
    grid:[
        {bottom:'55%'},
        {top:'55%'}
    ],

    series:[
        /*
        *第一个坐标系上的图表
        */
        {type:'bar',xAxisIndex:0,yAxisIndex:0},
        {type:'bar',xAxisIndex:0,yAxisIndex:0},
        {type:'bar',xAxisIndex:0,yAxisIndex:0},

        {type:'line',xAxisIndex:0,yAxisIndex:1},
        {type:'line',xAxisIndex:0,yAxisIndex:1},
        {type:'line',xAxisIndex:0,yAxisIndex:1},
        /*
        *第二个坐标系上的图表
        */

        //更改为by行的数据进行图表绘制的
        {type:'bar',xAxisIndex:1,yAxisIndex:2,seriesLayoutBy:'row'},
        {type:'bar',xAxisIndex:1,yAxisIndex:2,seriesLayoutBy:'row'},
```

```
        {type:'bar',xAxisIndex:1,yAxisIndex:2,seriesLayoutBy:'row'},
        {type:'bar',xAxisIndex:1,yAxisIndex:2,seriesLayoutBy:'row'},
    ]
}
myCharts.setOption(option);
```



维度映射

数据集的维度指的就是每个系列的名称name。

维度映射作用：对数据的维度信息统一定义和管理。

ECharts 默认会从 dataset.source 的第一行中获取维度名称。

但是，如果在dataset 里指定了 dimensions，那么 ECharts 不再会自动从 dataset.source 中获取维度信息。

```
/*dimensions中定义维度集合，值如下
*   null：不为此处维度作定义
*   {type: 'ordinal'}：只定义维度类型，type 有以下几种类型
*       number：默认，表示普通数字
*       ordinal：离散型，一般文本使用这种类型，echarts 会自动判断此类型。
*       float：Float64Array 浮点型
*       int：Int32Array 整形
*   {name: 'good', type: 'number'}：维度名称、维度类型都有
*   'bad' ：只指定维度名称，等同于 {name: 'bad'}
*/
dimensions: [ null,  {type: 'ordinal'}, {name: 'good', type: 'number'}, 'bad' ]
```

维度映射步骤:

1.通过dimensions设置维度集

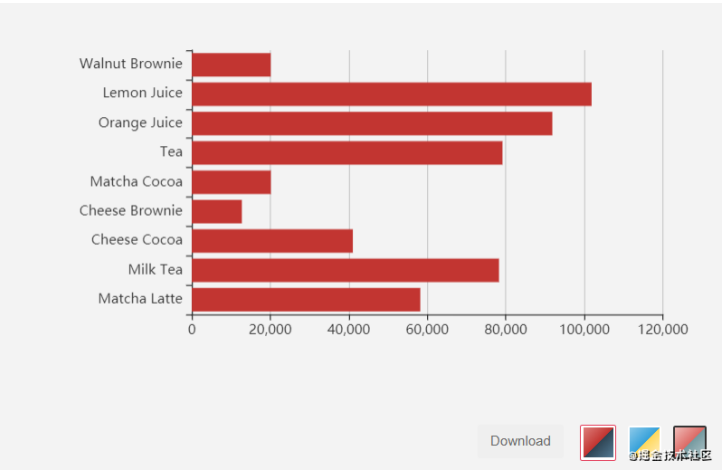
2.加入到dataset对象中

```
/*
案例：数据源未定义维度信息，通过dimensions设置维度集
*/

//数据源
const source=[
    ['html',  20,   25],
    ['css',   10,   15],
    ['js',    30,   40]
];
//维度集 dimensions
const dimensions=['大前端',null, '就业人数'];
// const dimensions=['大前端',{name:'学习人数'}, '就业人数'];
// const dimensions=['大前端','学习人数', '就业人数'];

// 指定图表的配置项和数据
const option = {
    legend: {},
    tooltip: {},
    dataset: {source,dimensions},
    xAxis: {},
    yAxis: {type: 'category'},
    series: [
        {type: 'bar'},
        {type: 'bar'},
    ]
};
```

编码映射

编码映射就是将x/y轴换成我们想要的维度的数据,通过下面的案例可以感受下什么是编码映射

```
var option = {
    dataset: {
        source: [
            ['score', 'amount', 'product'],
            [89.3, 58212, 'Matcha Latte'],
            [57.1, 78254, 'Milk Tea'],
            [74.4, 41032, 'Cheese Cocoa'],
            [50.1, 12755, 'Cheese Brownie'],
            [89.7, 20145, 'Matcha Cocoa'],
            [68.1, 79146, 'Tea'],
            [19.6, 91852, 'Orange Juice'],
            [10.6, 101852, 'Lemon Juice'],
            [32.7, 20112, 'Walnut Brownie']
        ]
    },
    grid: {containLabel: true},
    xAxis: {},
    yAxis: {type: 'category'},
    series: [
        {
            type: 'bar',
            encode: {
                // Map the "amount" column to X axis.
                x: 'amount',
                // Map the "product" column to Y axis
                y: 'product'
            }
        }
    ]
}
```

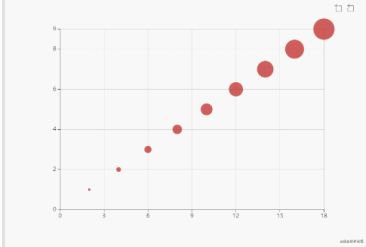
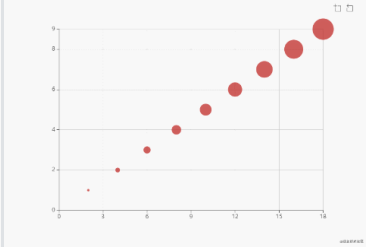
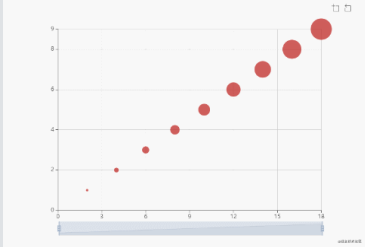


区域缩放

作用：概览整体，观察细节

区域缩放的方式：

- 框选型数据区域缩放组件（dataZoomSelect）：提供一个选框进行数据区域缩放。即 toolbox.feature.dataZoom，配置项在 toolbox 中。
- 内置型数据区域缩放组件（dataZoomInside）：内置于坐标系中，使用户可以在坐标系上通过鼠标拖拽、鼠标滚轮、手指滑动（触屏上）来缩放或漫游坐标系。
- 滑动条型数据区域缩放组件（dataZoomSlider）：有单独的滑动条，用户在滑动条上进行缩放或漫游

框选型数据区域缩放组件 (dataZoomSelect)	内置型数据区域缩放组件 (dataZoomInside)	滑动条型数据区域缩放组件 (dataZoomSlider)
`toolbox:{feature: {dataZoom: {}}}`	`dataZoom: [{type:'inside'}]`	`dataZoom: [{type:'slider'}]`
		

```
//数据源
const source = [
    //x   y   z
    [2,  1, 5],
    [4,  2, 10],
    [6,  3, 15],
    [8,  4, 20],
    [10, 5, 25],
    [12, 6, 30],
    [14, 7, 35],
    [16, 8, 40],
    [18, 9, 45],
```

```
];

// 指定图表的配置项和数据
const option = {
    tooltip: {},
    /*工具栏 toolbox
    *   feature{} 工具配置项
    *     dataZoom{} 框选型缩放缩放
    * */
    toolbox:{
        feature:{
            dataZoom: {}
        }
    },
    /*
    * x 轴
    *   min 最小值
    *       dataMin 取所有数据中的最小值
    *   max 最大值
    *       dataMax 取所有数据中的最大值
    * */
    xAxis: {
        type: 'value',
        min: 'dataMin',
        max: 'dataMax',
    },
    yAxis: {
        type: 'value',
        min: 'dataMin',
        max: 'dataMax',
    },
    /*
    * dataZoom 区域缩放 [{},{}]
    *   type 缩放方式
    *       inside 内置缩放，通过鼠标的平移缩放实现
    *       slider 滑动条缩放
    *   xAxisIndex 缩放作用于哪个x轴
    *   yAxisIndex 缩放作用于哪个y轴
    *   start 起始位，百分百 [0,100]
    *   end 结束位，百分百 [0,100]
    * */
    dataZoom:[
        {
            type:'inside',
            // start:50,
            // end:80,
        },
        // {
        //     type:'slider',
        //     // start:50,
        //     // end:80,
        //     xAxisIndex:0
        // },
        // {
        //     type:'slider',
        //     // start:50,
        //     // end:80,
        //     yAxisIndex:0
```

```
        // },
    ],

    /*数据集*/
    dataset:{source},
    /*系列列表*/
    series: [
        {
            type: 'scatter',
            symbolSize: function (param) {
                console.log(param)
                return param[2];
            },
        },
    ]
};
```

视觉映射

visualMap 视觉映射可以让项目的数据和颜色、大小等属性相关联。

举个例子：

source = [

[1, 1, **5**],

[2, 2, **9**]

]

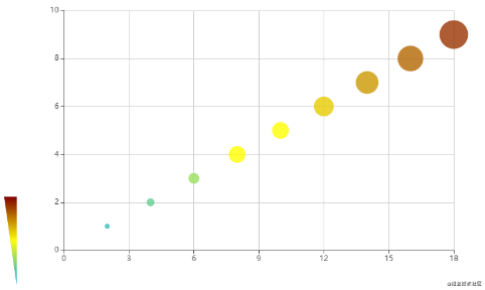
数据源source 的第一列和第二列分别对应散点在直角坐标系中的x、y 信息，第三列则默认对应 visualMap 。

若果我设置一个从绿色到红色的渐变区间，那么1 就对应绿色，9 就对应红色。

注意：visualMap 以前叫dataRange，如果你看到了比较老的教程或博客，里面有dataRange，要知道那就是视觉映射 visualMap

```
* visualMap 视觉映射 {}
*   type 映射方式
*       continuous 连续型
*       piecewise 分段型
*   min 映射区间的起始位置，如0
*   max 映射区间的接收位置，如90
*   calculable 是否显示拖拽用的手柄，只适用于连续型
*   range [] 显示此范围内的项目，百分比类型，只适用于连续型,如[0,100]
*   dimension 基于哪个维度的数据进行视觉映射
*   inRange 自定义取色范围
*       color[] 颜色映射
*       symbolSize[] 大小映射
*
* */
visualMap:{
    type:'continuous',
```

```
    min:0,
    // max:10,
    max:100,
    // dimension:1,
    //10/(100-0)=0.1
    inRange:{
        color:['#00acec','yellow','maroon'],
        //0.1*(50-3)+3
        symbolSize:[3,50]
    }
    // show:false
},
```



事件

鼠标事件

- ECharts 使用**on** 绑定事件，事件名称对应 DOM 事件名称，均为小写的字符串
- ECharts 支持常规的鼠标事件类型，包括 `'click'、'dblclick'、'mousedown'、'mousemove'、'mouseup'、'mouseover'、'mouseout'、'globalout'、'contextmenu'` 事件。
- 所有的鼠标事件包含参数 params，类似于dom事件的event

```
//点击图表中系列触发事件
myChart.on('click', function (params) {
    // 控制台打印数据的名称
    console.log(params);
});
//图例开关的行为会触发 legendselectchanged
myChart.on('legendselectchanged', function (params) {
        // 获取点击图例的选中状态
        let isSelected = params.selected[params.name];
        // 在控制台中打印
        console.log((isSelected ? '选中了' : '取消选中了') + '图例' + params.name);
        // 打印所有图例的状态
        console.log(params.selected);
});
```

调用 dispatchAction 后触发的事件

触发echarts图表行为，例如图形的高亮`highlight`，数据区域缩放`dataZoom`，显示提示框`showTip`

```
mycharts.dispatchAction({
    type: 'highlight',
    // 可选，系列 index，可以是一个数组指定多个系列
    seriesIndex?: number|Array,
    // 可选，系列名称，可以是一个数组指定多个系列
    seriesName?: string|Array,
    // 可选，数据的 index
    dataIndex?: number,
    // 可选，数据的 名称
    name?: string
})
```

富文本标签

富文本标签，就是内容丰富的文本标签。可以在各处的 `rich` 属性中定义文本片段样式。例如 series-bar.label.rich

富文本的实现步骤

1.用formatter 写文本片段

```
formatter:
   '{a|文字内容}\n'+
   '{b|文字内容}\n'+
   '默认样式{x|样式 x}'
 相当于：
<span class="a">样式a</span>\n

<span class="b">样式b</span>\n

默认样式 <span class="x">x</span>
```

2.用rich 设置文本样式

```
// 基于准备好的dom，初始化echarts实例
const myChart = echarts.init(document.getElementById('main'));

// 数据
const data=[
    {name:'杨戬',value:80,img:'./images/yj.jpg'},
    {name:'鲁班',value:60,img:'./images/lb.jpg'},
    {name:'沈梦溪',value:40,img:'./images/smx.jpg'},
    {name:'诸葛亮',value:30,img:'./images/zgl.jpg'}
];
data.forEach(item=>{
    setRich(item);
})

function setRich(item){
    /*自定义标签 label
*   formatter 文本片段
*       '{样式名|文字内容}\n 换行'
*   文本块的样式
*       textBorderColor 文本描边颜色
*       textBorderWidth 文本描边宽度
*       ...
```

```
*   rich 富文本，在其中写入样式
*       width 宽
*       height 高
*       backgroundColor 背景色
*           image 背景图
*       fontSize 文字大小
*       lineHeight 行高
*       fontWeight 文本加粗
*       ...
* */
    item.label={
        textBorderColor:'#fff',
        textBorderWidth:4,
        formatter:'{img|}\n{name|'+item.name+'}\n{val|战力: '+item.value+'}',
        rich:{
            img:{
                width:60,
                height:60,
                backgroundColor:{
                    image:item.img
                }
            },
            name:{
                fontSize:18,
                lineHeight:32
            },
            val:{
                fontSize:14
            },
        }
    }
}

/*配置项*/
const option = {
    title:{text:'英雄战力'},
    series: {
        type: 'pie',
        data,
        radius:'70%',
    }
};
myChart.setOption(option);
```

英雄战力