echarts高级应用

李伟

课堂目标

- 深度认知echarts
- 提高对复杂图表项目的开发能力

知识点综述

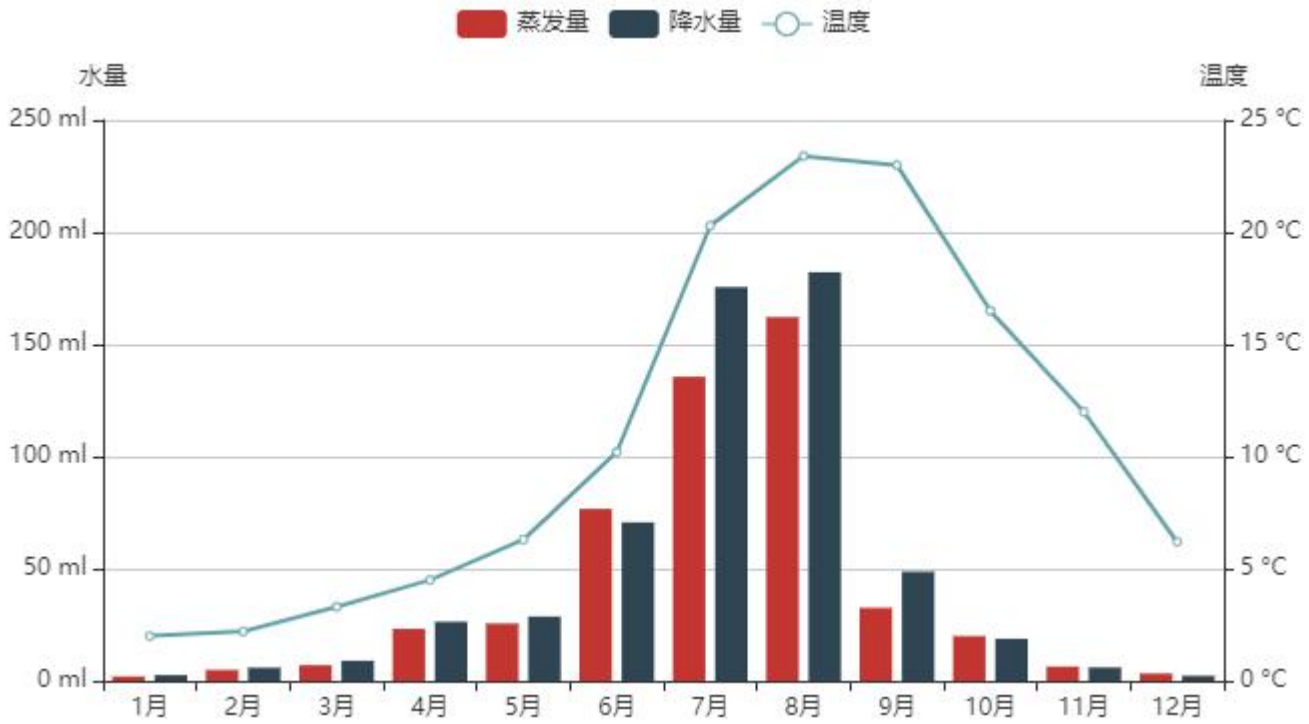
1. 多坐标轴
2. 异步数据
3. 数据集
4. 区域缩放
5. 视觉映射
6. 事件
7. 富文本标签
8. 原生图形组件
9. 响应式布局

多坐标轴

多坐标轴的常见应用就是一个坐标系有两个y 轴。

多坐标轴的设置方法：

1. 在yAxis 中写入两组数据，让两组数据的行数保持一致
2. 在series 中设置数据时，使用yAxisIndex 属性设置系列与哪个y 轴相关联



多坐标轴 - 让两组数据的行数保持一致

对于这个问题，我们需要手动设置每组数据的最大值、最小值和行间距。

通过最大值、最小值的设置，我们会的的得到一个数据展示的区间，这个区间是有长度的。

将区间长度除以行间距，就可以得到行数。

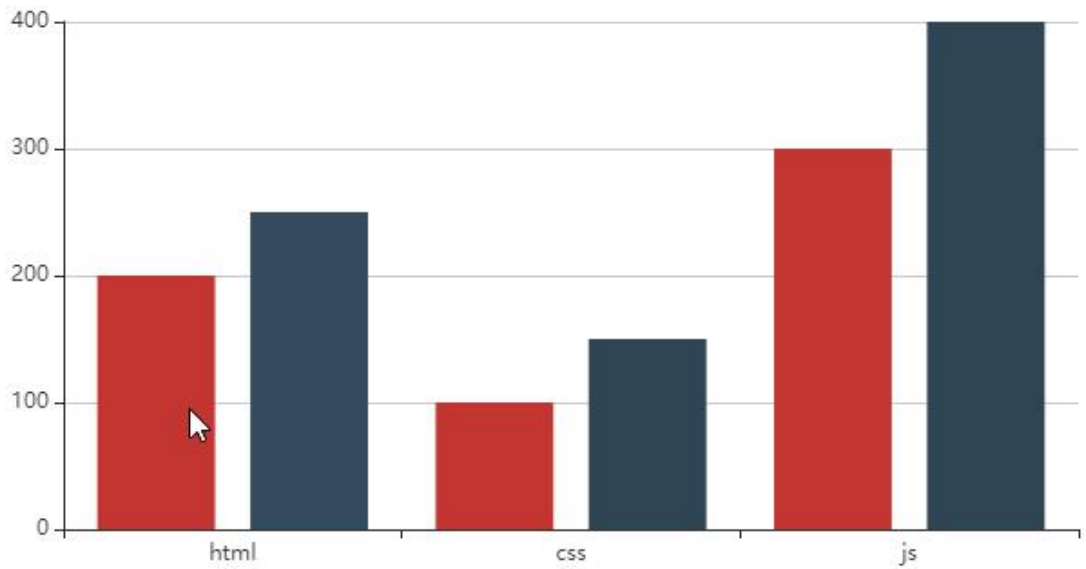
两组数据的行数要保持一致。

```
yAxis:[
    {
        axisLabel:{formatter:'{value} ml' },
        min:0,
        max:40,
        interval:8,
    },
    {
        axisLabel:{formatter:'{value} °C' },
        min:0,
        max:10,
        interval:2,
    }
],
```

类目轴和数值轴的概念

xAxis.type 默认是类目轴 category，适用于离散的类目数据，为该类型时必须通过 data 设置类目数据。

yAxis.type 默认是数值轴 value，适用于连续数据。



数据更新

对于请求数据的方式，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：

1. 请求到数据后，setOption()
2. 先setOption()，也就是有什么配置什么。请求到数据后，再追加配置

注：在数据加载的过程中，还可以使用loading

- 显示 loading：showLoading()
- 隐藏 loading：hideLoading()

数据集 dataset

dataset 数据集组件是从ECharts 4 开始有的，用于数据管理。

将数据写在每个series 系列中的方法，有以下缺陷：

- 不适合数据处理
- 不利于多个系列共享一份数据
- 不利于基于原始数据进行图表类型、系列的映射安排

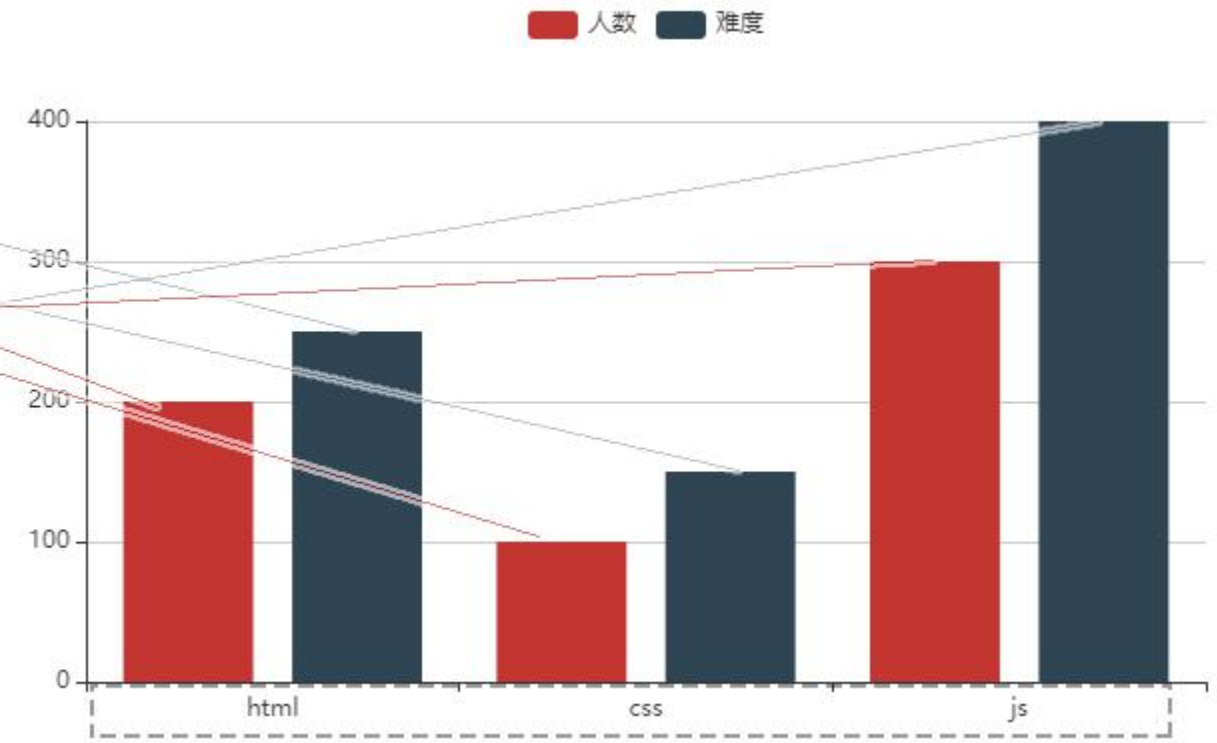
dataset 的优点：

- 基于数据，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。

数据集的理解方式

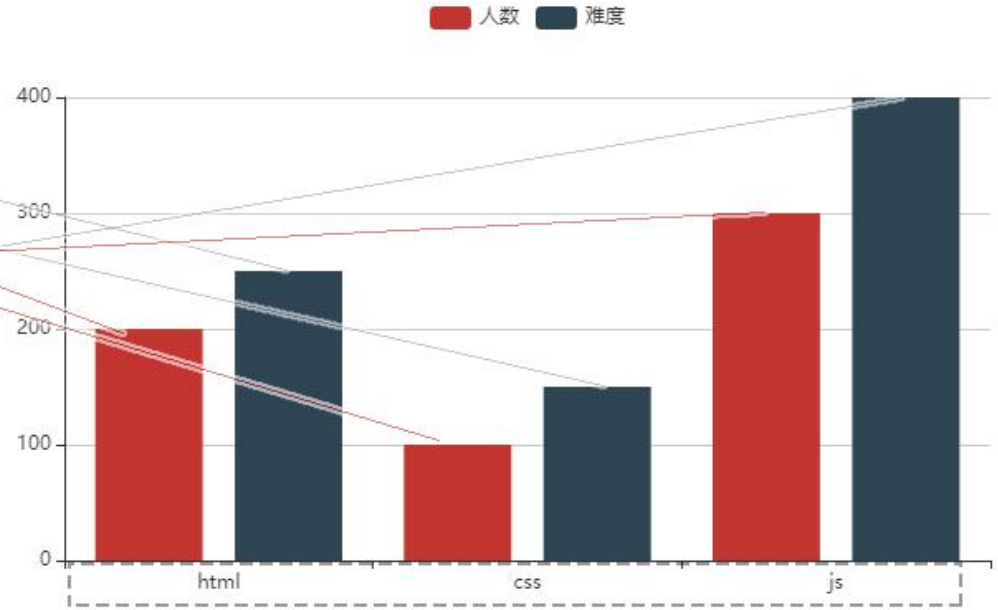
```
dataset: {
    // 提供一份数据。
    source: [
        ['大前端', '人数', '难度'],
        ['html',   200,    250],
        ['css',    100,    150],
        ['js',     300,    400]
    ]
},

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



数据集也可以使用对象数组

```
dataset: {
    // 提供一份数据。
    source: [
        ['大前端', '人数', '难度'],
        ['html',   200,    250],
        ['css',    100,    150],
        ['js',     300,    400]
    ]
},
series: [
    {type: 'bar'},
    {type: 'bar'},
]
```




```
dataset: {
    source: [
        {'大前端': 'html', '人数': 200, '难度': 250},
        {'大前端': 'css',  '人数': 100, '难度': 150},
        {'大前端': 'js',   '人数': 300, '难度': 400},
    ]
},
```

数据集的行列映射

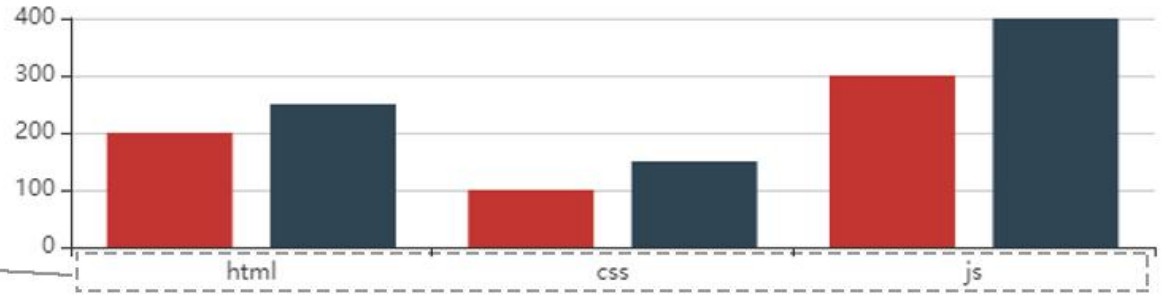
seriesLayoutBy：行列映射方式，会影响系列的划分方式

- column：基于列映射，默认
- row：基于行映射

数据集的行列映射

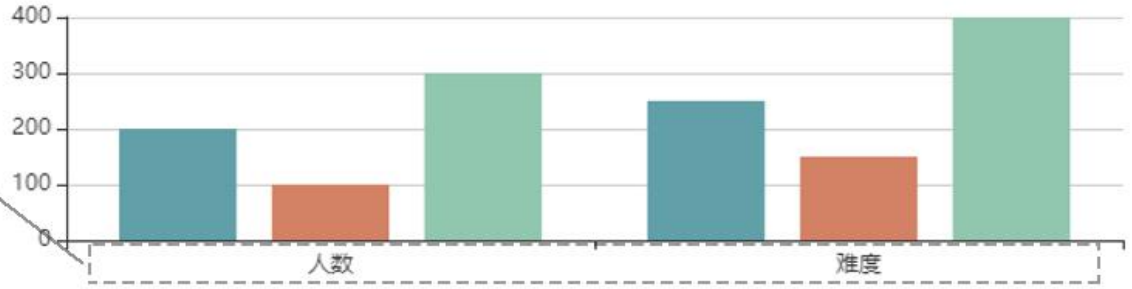
seriesLayoutBy：column

```
dataset: {
    // 提供一份数据。
    source: [
        ['大前端','人数', '难度'],
        ['html',   200,    250],
        ['css',    100,    150],
        ['js',     300,    400]
    ]
},
```



seriesLayoutBy：row

```
dataset: {
    // 提供一份数据。
    source: [
        ['大前端','人数', '难度'],
        ['html',   200,    250],
        ['css',    100,    150],
        ['js',     300,    400]
    ]
},
```



数据集的dimensions 维度映射

作用：定义 series.data 或者 dataset.source 的每个维度的信息。

ECharts 会自动从 dataset.source 的第一行/列中获取维度信息。

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

```
dataset: {
    dimensions:['大前端','人数', '难度'],
    source: [
        // ['大前端','人数', '难度'],
        ['html',   200,    250],
        ['css',    100,    150],
        ['js',     300,    400]
    ]
},
```

dimensions 中元素的书写方式

- 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’}

如：

```
dimensions: [ null,  {type: 'ordinal'}, {name: 'good', type: 'number'}, 'bad' ]
```

dimensions 的书写位置

dimensions 可以在两个地方设置：

- 在dataset 中设置，作用于所有系列
- 在series 系列中设置，作用于其所在的系列

```
dataset: {
    dimensions:['大前端','人数', '难度'],
    source: [
        // ['大前端','人数', '难度'],
        ['html',  200,   250],
        ['css',   100,   150],
        ['js',    300,   400]
    ]
},
series: [
    {
        type: 'bar',
        dimensions:['大前端','人数', '难度']
    },
    {type: 'bar'},
]
```

数据集的encode 编码映射

encode 可以定义 data 的哪个维度被编码成什么。

如，这样的数据，我们用它只建一个系列的图表。默认series 里第一个系列对应的就是数据源里的第二列数据。后面的以此类推。

```
//维度映射
const dimensions=['大前端','人数', '难度'];
//数据源
const source =[
    ['html',   20,    25],
    ['css',    10,    15],
    ['js',     30,    40],
];
```

```
const option = {
    tooltip:{},
    dataset: {dimensions,source},
    /*设置类目轴和数值轴*/
    xAxis:{type:'category'},
    yAxis:{type:'value'},
    /*encode 编码映射*/
    series:[
        {type:'bar'}
    ]
};
```

可是，如果我们想让series 里第一个系列映射数据源里的第二列，而且还不想改变数据源(数据源是公共资源)，应该怎么办呢？

这就要用到编码映射。

encode 编码映射的使用方法

encode 常见属性：

- tooltip：['product', 'score‘]，提示信息
- seriesName：[1, 3]，系列名
- x：x 轴的数据映射
- y：y 轴的数据映射

更多属性请看文档

区域缩放 dataZoom

作用：概览整体，观察细节

dataZoom 组件的类型：

- 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）：提供一个选框进行数据区域缩放。即 toolbox.feature.dataZoom，配置项在 toolbox 中。

visualMap 视觉映射的原理

visualMap 视觉映射可以让项目的数据和项目的颜色、大小等属性形成映射关系。

举个例子：

source = [

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

[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原色区间，那么1 就对应绿色，9 就对应红色。

visualMap 的常见属性

- type 映射方式
  - continuous 连续型
  - piecewise 分段型
- min、max 颜色映射的区间，对应实际数据
- range 只显示此范围内的项目
- calculable 是否显示拖拽用的手柄，在连续型的颜色映射器中
- inRange 自定义选中范围中的视觉元素
  - color[] 颜色映射
  - symbolSize 大小映射
  - ...

视觉映射 visualMap

注：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

ECharts 如何监听事件

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。如：

```
myChart.on('click', function (params) {
    // 控制台打印数据的名称
    console.log(params.name);
});
```

鼠标事件有哪些

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，如被点击的图形信息 params.componentType 。

如何监听组件交互事件

在 ECharts 中基本上所有的组件交互行为都会触发相应的事件，常用的事件和事件对应参数在 events 文档中有列出。

```
// 图例开关的行为只会触发 legendselectchanged 事件
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);
});
```

代码触发 ECharts 中组件的行为

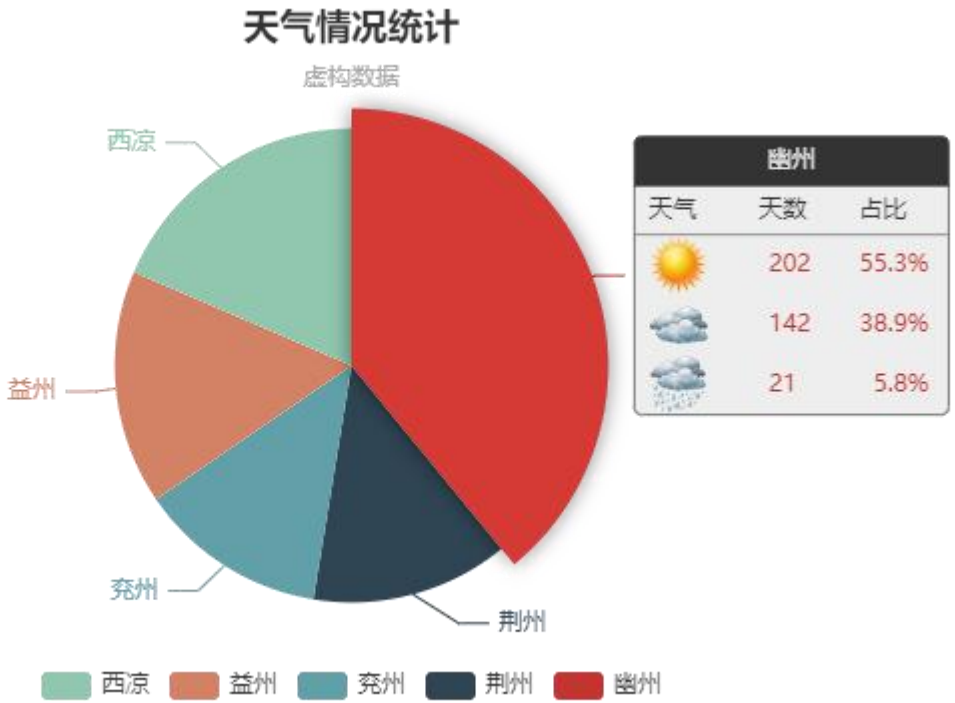
ECharts 通过调用 myChart.dispatchAction({ type: ‘’ }) 触发图表行为，如：

```
myChart.dispatchAction({
    type: 'highlight',
    seriesIndex: 0,
    dataIndex: app.currentIndex
});
```

富文本标签

富文本标签，就是内容丰富的文本标签。

在许多地方（如图、轴的标签等）都可以使用富文本标签。例如：



文本块和文本片段

- 文本块（Text Block）：文本标签块整体。
- 文本片段（Text fragment）：文本标签块中的部分文本。

文本标签的属性可以参考label：https://www.echartsjs.com/zh/option.html#series-bar.label

富文本的属性：https://www.echartsjs.com/zh/option.html#series-bar.label.rich.%3Cstyle_name%3E

富文本的实现步骤

1. 首先要确定一个承载富文本的载体，比如提示、标签等
2. 要有formatter，这个东东可以理解为html 标签，只不写法是完全不一样的。如：

```
formatter:
        '{a|样式 a}\n'+
        '{b|样式 b}\n'+
        '默认样式{x|样式 x}'
```

就相当于：

```
<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>
```

富文本的实现步骤

3.设置文本样式

```
rich: {
        a: {
            color: 'red',
        },
        b: {
            backgroundColor: {
                image: 'xxx/xxx.jpg'
            },
            height: 40
        },
        x: {
            fontSize: 18,
        },
    }
```

原生图形组件的基本概念

原生图形组件就是可以自己绘制图形的组件。

原生图形组件里绘制的图形，可以绑定鼠标事件、拖拽事件等。

echarts 有两种点位：坐标位，像素位。

坐标系点位有直角坐标位、地理坐标位等。

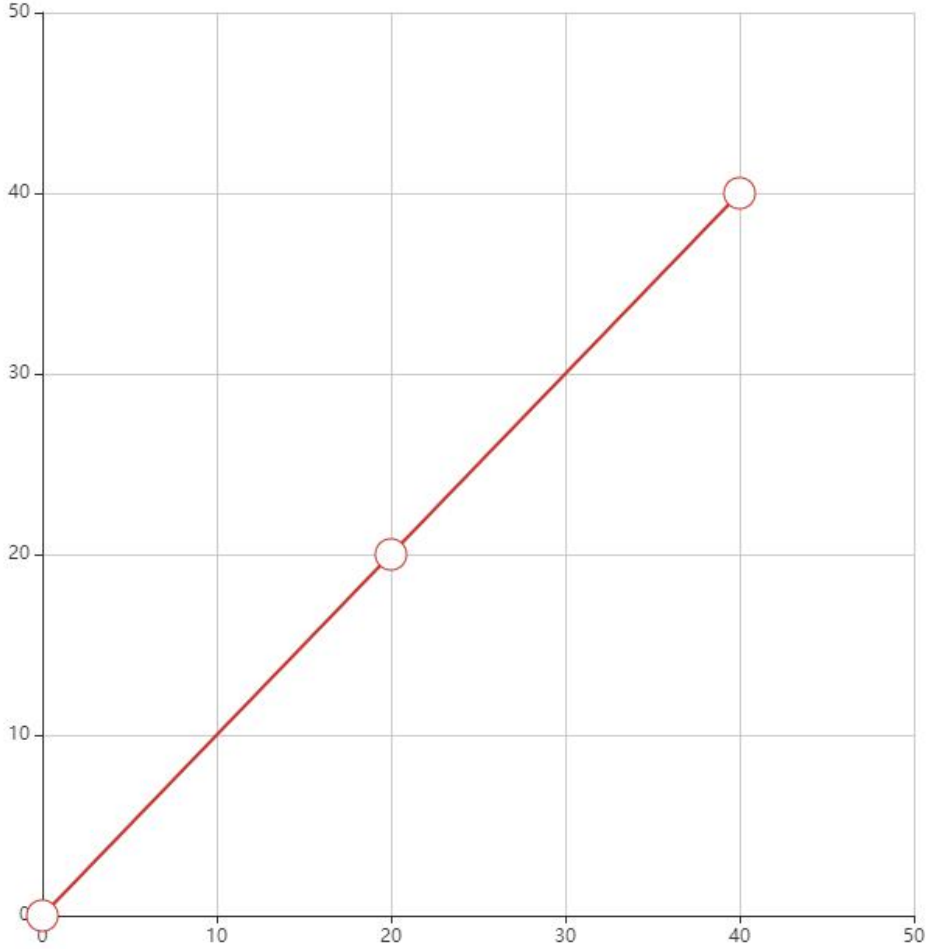
原生图形的位置就是基于像素位定位的。

echarts 实例对象提供了坐标位和像素位的转换方法：

- convertToPixel(坐标系，[x,y]) 坐标位转像素位
- convertFromPixel(坐标系，[x,y]) 像素位转坐标位

案例 – 折线图标记点的拖拽

1. 正常绘制折线图
2. 在折线图的所有标记点位置绘制原生图形
3. 为原生图形绑定鼠标事件：
   - 拖拽时，将原生图形的位置赋予标记点
   - 鼠标移动时，显示提示
   - 鼠标抬起时，隐藏提示



响应式布局

在html 中使用css 的flex 可以轻松实现响应式布局。

在echarts 里，如何适配不同尺寸的屏幕呢？

- 简单点的可以通过为尺寸、位置等属性设置百分比来实现。
- 复杂些的就需要自定义响应规则。

接下来咱们就是重点说一下自定义响应规则的方法。

自定义响应规则的方法

1. 建立基础配置项 baseOption
2. 建立规则配置列表 media

- 建立query
- 建立此规则下的配置信息option

3. echarts 实例基于baseOption 和media 绘制图表

总结

在我们实际项目开发中，肯定会遇到各种各样的需求。我们这章所说的都是出现频率较高的知识点。若想熟练灵活的掌控echarts，大家要可以基于echarts 的核心功能，多去阅读官方文档和案例。