1.ECharts高级

1.1.显示相关

1.1.1.主题

- 默认主题

  `ECharts`中默认内置了两套主题: `light` `dark`

  在初始化对象方法`init`中可以指明

  ```
  var chart = echarts.init(dom, 'light')
  var chart = echarts.init(dom, 'dark')
  ```

- 自定义主题
  - 1.在主题编辑器中编辑主题

    主题编辑器的地址为: `https://www.echartsjs.com/theme-builder/`

    在该地址中, 你可以定义一个主题的很多方面的内容:

    基本配置
    视觉映射
    坐标轴
    图例
    工具箱
    提示框
    时间轴
    数据缩放
    折线图
    K 线图
    力导图

  - 2.下载主题, 是一个`js`文件

    在线编辑完主题之后, 可以点击下载主题按钮, 下载主题的js文件

- 3.引入主题 `js` 文件

```
<script src="js/echarts.min.js"></script>
<script src="js/itcast.js"></script>
```

其中, `itcast.js` 就是下载下来的主题文件

- 4.在 `init` 方法中使用主题

```
var mCharts = echarts.init(document.querySelector("div"), 'itcast')
```

init方法中的第二个参数itcast就是主题的名称, 这个名称叫什么我们可以在itcast.js的代码中看出

```
}(this, function (exports, echarts) {
    var log = function (msg) {
        if (typeof console !== 'undefined') {
            console && console.error && console.error(msg);
        }
    };
    if (!echarts) {
        log('ECharts is not Loaded');
        return;
    }
    echarts.registerTheme('itcast', {
        "color": [
```

1.1.2.调色盘

它是一组颜色，图形、系列会自动从其中选择颜色, 不断的循环从头取到尾, 再从头取到尾, 如此往复.

- 主题调色盘

```
echarts.registerTheme('itcast', {
    "color": [
        "#893448",
        "#d95850",
        "#eb8146",
        "#ffb248",
        "#f2d643",
        "#ebdba4"
    ],
    "backgroundColor": "rgba(242,234,191,0.15)",
    ......
})
```

- 全局调色盘

  全局调色盘是在`option`下增加一个`color`的数组

```
var option = {
    // 全局调色盘
    color: ['red', 'green', 'blue'],
    ......
}
mCharts.setOption(option)
```

- 局部调色盘

  局部调色盘就是在`series`下增加一个`color`的数组

```
var option = {
    // 全局调色盘
    color: ['red', 'green', 'blue'],
    series: [
        {
            type: 'pie',
            data: pieData,
            // 局部调色盘
            color: ['pink', 'yellow', 'black']
        }
    ]
}
mCharts.setOption(option)
```

  需要注意一点的是, 如果全局的调色盘和局部的调色盘都设置了, 局部调色盘会产生效果, 这里面遵循的是就近原则

- 渐变颜色的实现

  在`ECharts`中, 支持线性渐变和径向渐变两种颜色渐变的方式

  - 线性渐变

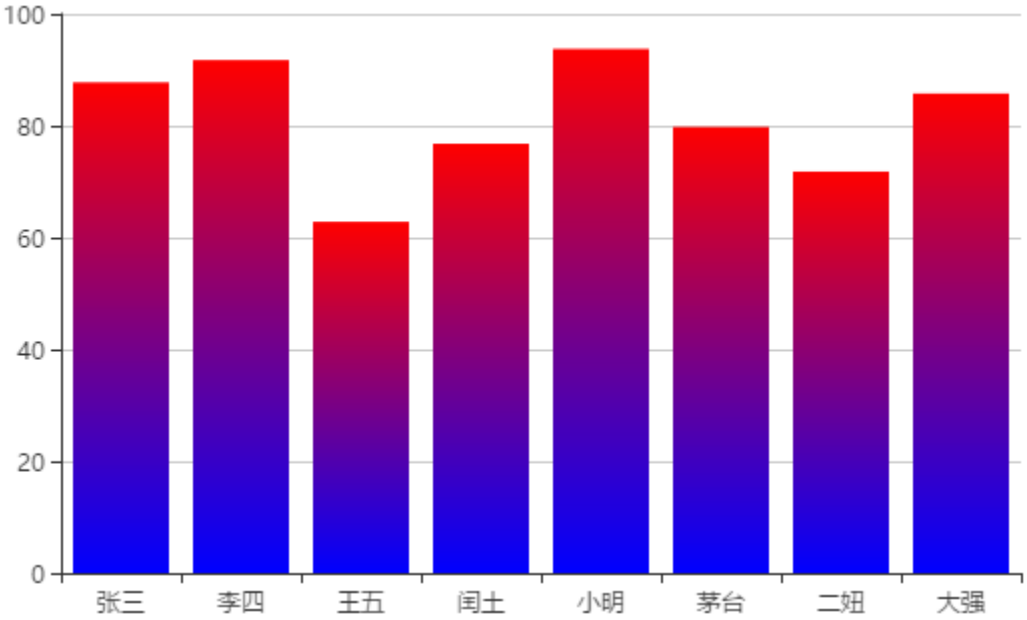
    线性渐变的类型为`linear`, 他需要配置线性的方向, 通过`x, y, x2, y2`即可进行配置

    `x`, `y`, `x2`, `y2`, 范围从 0 - 1，相当于在图形包围盒中的百分比，如果 `global` 为 `true`，则该四个值是绝对的像素位置

    在下述代码中的 `0 0 0 1` 意味着从上往下进行渐变

```
<!DOCTYPE html>
```

```
<html lang="en">
<head>
  <script src="js/echarts.min.js"></script>
</head>
<body>
  <div style="width: 600px;height:400px"></div>
  <script>
  var mCharts = echarts.init(document.querySelector("div"))
  var xDataArr = ['张三', '李四', '王五', '闰土', '小明', '茅台', '二妞', '大
强']
  var yDataArr = [88, 92, 63, 77, 94, 80, 72, 86]
  var option = {
    xAxis: {
      type: 'category',
      data: xDataArr
    },
    yAxis: {
      type: 'value'
    },
    series: [
      {
        type: 'bar',
        data: yDataArr,
        itemStyle: {
          color: {
            type: 'linear',
            x: 0,
            y: 0,
            x2: 0,
            y2: 1,
            colorStops: [{
              offset: 0, color: 'red' // 0% 处的颜色
            }, {
              offset: 1, color: 'blue' // 100% 处的颜色
            }],
            globalCoord: false // 缺省为 false
          }
        }
      }
    ]
};
  mCharts.setOption(option)
    </script>
</body>
</html>
```



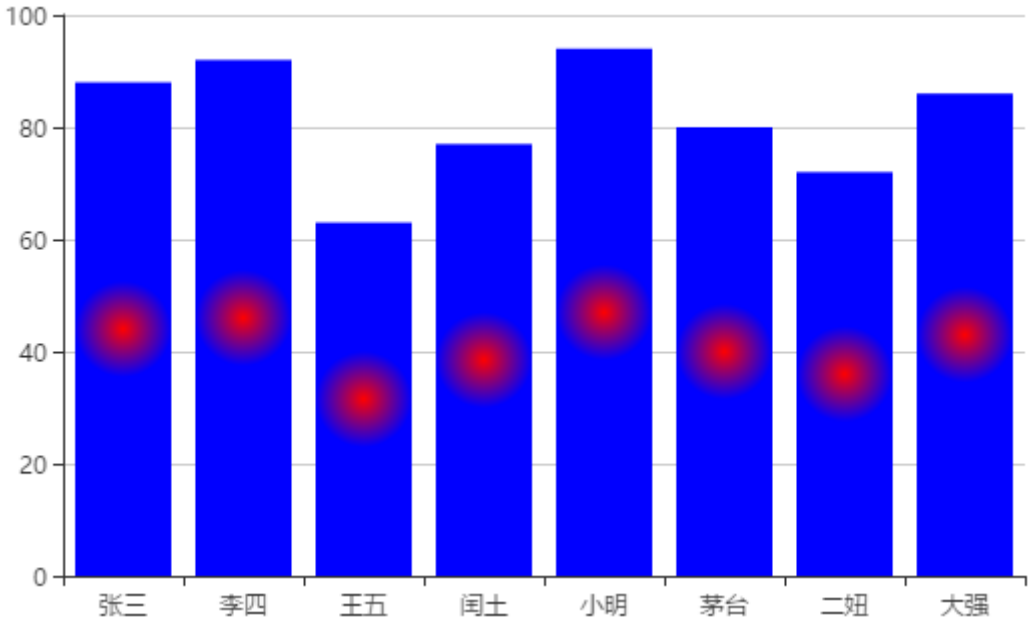
- 径向渐变

  线性渐变的类型为`radial`, 他需要配置径向的方向, 通过`x`, `y`, `r`即可进行配置

  前三个参数分别是圆心 `x`, `y` 和半径，取值同线性渐变

  在下述代码中的 0.5 0.5 0.5 意味着从柱的重点点, 向外径向扩散半径为宽度一半的圆

```
series: [
      {
          itemStyle: {
              color: {
                  type: 'radial',
                  x: 0.5,
                  y: 0.5,
                  r: 0.5,
                  colorStops: [{
                      offset: 0, color: 'red' // 0% 处的颜色
                  }, {
                      offset: 1, color: 'blue' // 100% 处的颜色
                  }],
                  global: false // 缺省为 false
              }
          }
      }
  ]
```



1.1.3.样式

- 直接样式
  - `itemStyle`
  - `textStyle`
  - `lineStyle`
  - `areaStyle`
  - `label`

```
data: [
    {
        value: 11231,
        name: "淘宝",
        itemStyle: {
            color: 'black'
        }
    }
]
title: {
    text: '我是标题',
    textStyle: {
        color: 'red'
    }
}
label: {
    color: 'green'
}
```

这些样式一般都可以设置颜色或者背景或者字体等样式, 他们会覆盖主题中的样式

- 高亮样式

图表中, 其实有很多元素都是有两种状态的, 一种是默认状态, 另外一种就是鼠标滑过或者点击形成的高亮状态. 而高亮样式是针对于元素的高亮状态设定的样式

那它的使用也非常简单,在`emphasis`中包裹原先的`itemStyle`等等, 我们来试一下

```
series: [
```

```
    {
        type: 'pie',
        label: {
            color: 'green'
        },
        emphasis: {
            label: {
                color: 'red'
            },
        },

data: [{
    value: 11231,
    name: "淘宝",
    itemStyle: {
        color: 'black'
    },
    emphasis: {
        itemStyle: {
            color: 'blue'
        },
    }
},
```

1.1.4.自适应

- 步骤1: 监听窗口大小变化事件
- 步骤2: 在事件处理函数中调用 `ECharts` 实例对象的 `resize` 即可

```
<!DOCTYPE html>
<html lang="en">
<head>
    <script src="js/echarts.min.js"></script>
</head>
<body>
    <div style=" height:400px;border:1px solid red"></div>
    <script>
    var mCharts = echarts.init(document.querySelector("div"))
    var xDataArr = ['张三', '李四', '王五', '闰土', '小明', '茅台', '二妞', '大
强']
    var yDataArr = [88, 92, 63, 77, 94, 80, 72, 86]
    var option = {
        xAxis: {
            type: 'category',
            data: xDataArr
        },
        yAxis: {
            type: 'value'
        },
        series: [
            {
                type: 'bar',
                data: yDataArr
            }
        ]
    };
```

```
    mCharts.setOption(option)
    // 监听window大小变化的事件
    window.onresize = function () {
        // 调用echarts示例对象的resize方法
        mCharts.resize()
    }
    // window.onresize = mCharts.resize
    </script>
</body>
</html>
```

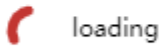
1.2.动画的使用

1.2.1.加载动画

`ECharts`已经内置好了加载数据的动画, 我们只需要在合适的时机显示或者隐藏即可

- 显示加载动画

```
mCharts.showLoading()
一般，我们会在获取图表数据之前  显示加载动画
```



- 隐藏加载动画

```
mCharts.hideLoading()
一般，我们会在获取图表数据之后  隐藏加载动画，显示图表
```

1.2.2.增量动画

所有数据的更新都通过`setOption`实现, 我们不用考虑数据到底产生了那些变化, `ECharts`会找到两组数据之间的差异然后通过合适的动画去表现数据的变化。

```
<!DOCTYPE html>
<html lang="en">
<head>
    <script src="js/echarts.min.js"></script>
</head>
<body>
    <div style="width: 600px;height:400px"></div>
```

```
    <button>修改数据</button>
    <button id="btnAdd">增加数据</button>
    <script>
    var mCharts = echarts.init(document.querySelector("div"))
    var xDataArr = ['张三', '李四', '王五', '闰土', '小明', '茅台', '二妞', '大强']
    var yDataArr = [88, 92, 63, 77, 94, 80, 72, 86]
    var option = {
        xAxis: {
            type: 'category',
            data: xDataArr
        },
        yAxis: {
            type: 'value'
        },
        series: [
            {
                type: 'bar',
                data: yDataArr
            }
        ]
    };
    mCharts.setOption(option)
    var btn = document.querySelector('button');
    btn.onclick = function () {
        var newArr = [68, 62, 93, 67, 64, 90, 52, 36]
        // setOption的方法可以被调用多次
        // 新的option 和旧的option配置
        // 新旧option配置项他们之间不是替换的关系,是相互整合的关系
        // 我们在设置新的option的时候,只需要考虑到将变化的配置项配置就可以了
        var option = {
            series: [
                {
                    data: newArr,
                }
            ]
        };
        mCharts.setOption(option)
    }

    var btnAdd = document.querySelector('#btnAdd')
    btnAdd.onclick = function () {
        setInterval(function () {
            //增加数据
            xDataArr.push('小明')
            yDataArr.push(parseInt(50 + Math.random() * 10))
            var option = {
                xAxis: {
                    data: xDataArr
                },
                series: [
                    {
                        data: yDataArr
                    }
                ]
            }
            mCharts.setOption(option)
        }, 1000)
    }
```

```
    </script>
</body>
</html>
```

1.2.3.动画的配置

- 开启动画

  `animation: true`

- 动画时长

  `animationDuration: 5000 或 回调函数`

  实现每根柱条依次缓动效果:

```
animationDuration: arg => {  // arg参数: 平均值 最值 x轴样本
    console.log(arg)
    return 2000 * arg // 使每根柱条的动画时长都不一样
},
```

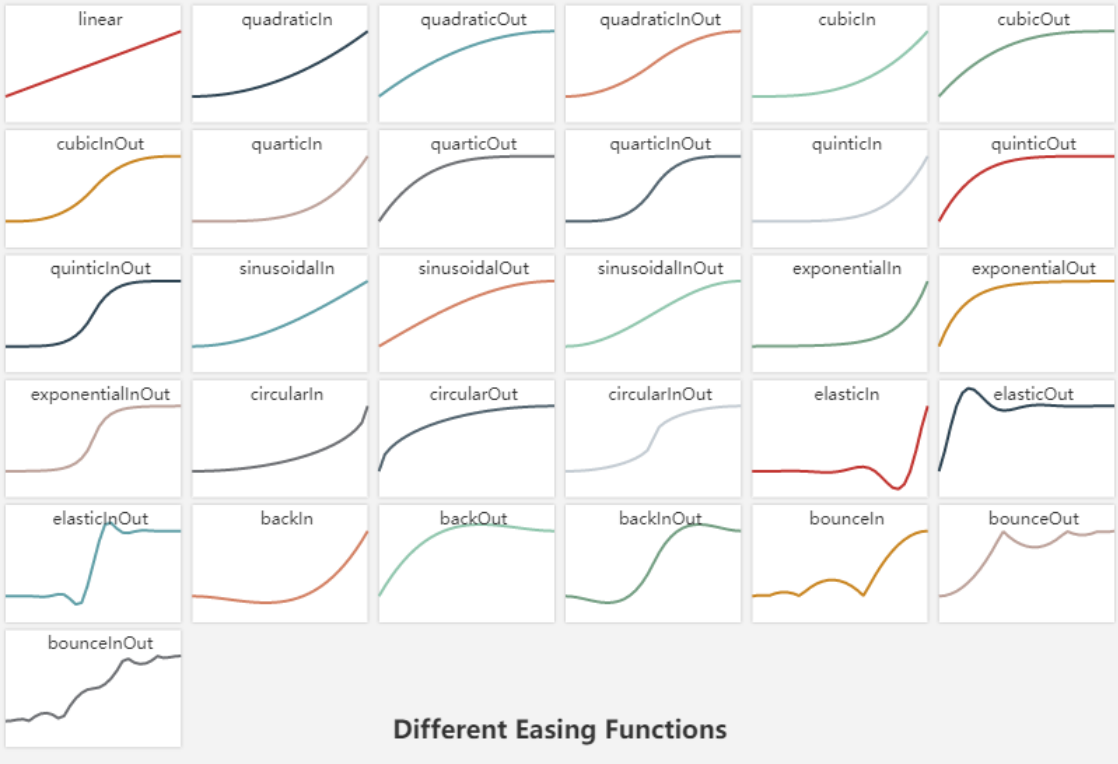
- 缓动动画

  `animationEasing: 'bounceOut'`

  `linear`,线性变化, 这样动画效果会很均匀

  `bounceOut`,这样动画效果会有一个回弹效果

  缓动动画的可选值如下图:



- 动画阈值

  `animationThreshold: 8`

  单种形式的元素数量大于这个阈值时会关闭动画

1.3.交互API

1.3.1.全局`echarts`对象

全局`echarts`对象是引入`echarts.js`文件之后就可以直接使用的

- `echarts.init`

初始化ECharts实例对象
使用主题

- `echarts.registerTheme`

注册主题
只有注册过的主题,才能在init方法中使用该主题

- `echarts.registerMap`

注册地图数据
```
$.get('json/map/china.json', function (chinaJson) {
            echarts.registerMap('china', chinaJson);
 });
```
geo组件使用地图数据
```
var option = {
    geo: {
          type: 'map',
          map: 'china',
    },
})
```

- `echarts.connect`
  - 一个页面中可以有多个独立的图表
  - 每一个图表对应一个`ECharts`实例对象
  - `connect`可以实现多图关联，传入联动目标为`EChart`实例，支持数组

```
<!DOCTYPE html>
<html lang="en">
<head>
    <script src="js/echarts.min.js"></script>
    <script src="js/jquery.min.js"></script>
</head>
<body>
    <div style="width: 600px;height:400px;border:1px solid red"></div>
    <div style="width: 600px;height:400px;border:1px solid green" id="div1">
</div>
    <script>
    var mCharts = echarts.init(document.querySelector("div"), 'itcast')
    var xDataArr = ['张三', '李四', '王五', '闰土', '小明', '茅台', '二妞', '大
强']
    var yDataArr = [88, 92, 63, 77, 94, 80, 72, 86]
    var option = {
        xAxis: {
            type: 'category',
            data: xDataArr
        },
        toolbox: {
            feature: {
                saveAsImage: {}
            }
        },
        yAxis: {
            type: 'value'
```

```
        },
        series: [
            {
                type: 'bar',
                data: yDataArr
            }
        ]
    };
    mCharts.setOption(option)

    $.get('json/map/china.json', function (chinaJson) {
        echarts.registerMap('china', chinaJson)
        var mCharts2 = echarts.init(document.querySelector('#div1'));
        var option2 = {
            geo: {
                type: 'map',
                map: 'china'
            }
        }
        mCharts2.setOption(option2)
        echarts.connect([mCharts, mCharts2])
    })
    </script>
</body>
</html>
```

这样, 由于我们打开了toolbox中的saveAsImage, 所以会出现下载图片的按钮. 而通过`echarts.connect([mCharts, mCharts2])`, 此时点击下载图片按钮, 保存下来的图片就是两个图表的图片了.

1.3.2. `echartsInstance`对象

`eChartsInstance`对象是通过`echarts.init`方法调用之后得到的

- `echartsInstance.setOption`

  设置或修改图表实例的配置项以及数据
  多次调用`setOption`方法
  　　合并新的配置和旧的配置
  　　增量动画

- `echartsInstance.resize`

  重新计算和绘制图表
  一般和`window`对象的`resize`事件结合使用

```
window.onresize = function(){
     myChart.resize();
}
```

- `echartsInstance.on` `echartsInstance.off`

  绑定或者解绑事件处理函数

  - 鼠标事件

```
常见事件: 'click'、'dblclick'、'mousedown'、'mousemove'、'mouseup' 等
事件参数 arg:  和事件相关的数据信息
mCharts.on('click', function (arg) {
    // console.log(arg)
    console.log('饼图被点击了')
})
解绑事件:
mCharts.off('click')
```

- `ECharts`事件

```
常见事件:
legendselectchanged、'datazoom'、'pieselectchanged'、'mapselectchanged'
等
事件参数 arg:  和事件相关的数据信息
mCharts.on('legendselectchanged', function (arg) {
    console.log(arg)
    console.log('图例选择发生了改变...')
})
```

- `echartsInstance.dispatchAction`

  主动触发某些行为, 使用代码模拟用户的行为

```
// 触发高亮的行为
mCharts.dispatchAction({
    type: "highlight",
    seriesIndex: 0,
    dataIndex: 1
})
// 触发显示提示框的行为
mCharts.dispatchAction({
    type: "showTip",
    seriesIndex: 0,
    dataIndex: 3
})
```

- `echartsInstance.clear`

  清空当前实例，会移除实例中所有的组件和图表

  清空之后可以再次`setOption`

- `echartsInstance.dispose`

  销毁实例

  销毁后实例无法再被使用