# confirmation.html

<!DOCTYPE html>  
<html lang="en">  
  
<head>  
 <meta charset="UTF-8">  
 <meta http-equiv="X-UA-Compatible" content="IE=edge">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <title>存证确权</title>  
 <style type="text/css">  
 a{text-decoration:none;}  
 </style>  
 <link rel="stylesheet" href="css/confirmation.css">  
 <script src=" /libs/jquery/2.1.4/jquery.min.js"></script>  
 <script src="js/flexible.js"></script>  
 <script src="js/echarts.min.js"></script>  
 <script src="js/confirmation\_index.js"></script>  
</head>  
  
<body>  
 <!-- 头部的盒子 -->  
 <header>  
 <h1>  
 <ul class="button-group">  
 <li><button class="large blue button" ><a href="confirmation.html" style="font-size: 0.4rem;color: white" >存证确权</a></button></li>  
 <li><button class="large blueShadow button" ><a href="surveillance.html" style="font-size: 0.4rem;color: white" >监测维权</a></button></li>  
 </ul>  
 </h1>  
 <div class="showTime"></div>  
 <script>  
 var ***t*** = null;  
 ***t*** = setTimeout(time, 1000);//開始运行  
 function time() {  
 clearTimeout(***t***);//清除定时器  
 dt = new ***Date***();  
 var y = dt.getFullYear();  
 var mt = dt.getMonth() + 1;  
 var day = dt.getDate();  
 var h = dt.getHours();//获取时  
 var m = dt.getMinutes();//获取分  
 var s = dt.getSeconds();//获取秒  
 ***document***.querySelector(".showTime").innerHTML = '当前时间：' + y + "年" + mt + "月" + day + "日-" + h + "时" + m + "分" + s + "秒";  
 ***t*** = setTimeout(time, 1000); //设定定时器，循环运行   
 }  
 </script>  
 </header>  
  
 <!-- 页面主体部分 -->  
 <section class="mainbox">  
 <div class="column">  
  
 <div class="panel upperleft">  
 <h2>新增不同作品类型的存证数量/季</h2>  
 <div class="chart">图表 </div>  
 <div class="panel-footer"></div>  
 </div>  
  
 <section class="innerbox">  
 <div class="innercolumn">  
 <div class="innerpanel lowerleftleft">  
 <h2>新增存证数量/月</h2>  
 <div class="chart">图表 </div>  
 <div class="innerpanel-footer"></div>  
 </div>  
 </div>  
 <div class="innercolumn"></div>  
 <div class="innercolumn">  
 <div class="innerpanel lowerleftright">  
 <h2>新增版权通证数量/月</h2>  
 <div class="chart">图表 </div>  
 <div class="innerpanel-footer"></div>  
 </div>  
 </div>  
 </section>  
 </div>  
  
 <div class="column">  
 <div class="rightpanel upperright">  
 <h2>当前不同作品类型存证数量分布</h2>  
 <div class="chart">图表 </div>  
 <div class="rightpanel-footer"></div>  
 </div>  
 <div class="rightpanel middleright">  
 <h2>个人账户与非个人账户接收者通证数量分布</h2>  
 <div class="chart">图表 </div>  
 <div class="rightpanel-footer"></div>  
 </div>  
 <div class="rightpanel lowerright">  
 <h2>不同类别通证数量分布</h2>  
 <div class="chart">图表 </div>  
 <div class="rightpanel-footer"></div>  
 </div>  
 </div>  
 </section>  
   
</body>  
  
</html>

# surveillance.html

<!DOCTYPE html>  
<html lang="en">  
  
<head>  
 <meta charset="UTF-8">  
 <meta http-equiv="X-UA-Compatible" content="IE=edge">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <title>监测维权</title>  
 <link rel="stylesheet" href="css/surveillance.css">  
</head>  
  
<body>  
 <!-- 头部的盒子 -->  
 <header>  
 <h1>中控台/存证确权/监测维权/智能交易（待定模块）</h1>  
 <div class="showTime"></div>  
 <script>  
 var ***t*** = null;  
 ***t*** = setTimeout(time, 1000);//開始运行  
 function time() {  
 clearTimeout(***t***);//清除定时器  
 dt = new ***Date***();  
 var y = dt.getFullYear();  
 var mt = dt.getMonth() + 1;  
 var day = dt.getDate();  
 var h = dt.getHours();//获取时  
 var m = dt.getMinutes();//获取分  
 var s = dt.getSeconds();//获取秒  
 ***document***.querySelector(".showTime").innerHTML = '当前时间：' + y + "年" + mt + "月" + day + "日-" + h + "时" + m + "分" + s + "秒";  
 ***t*** = setTimeout(time, 1000); //设定定时器，循环运行   
 }  
 </script>  
 </header>  
  
 <!-- 页面主体部分 -->  
 <section class="mainbox">  
 <div class="column">  
 <div class="panel bar" >  
 <h2>柱形图-1 </h2>  
 <div class="chart">图表 </div>  
 <div class="panel-footer"></div>  
 </div>  
 <div class="panel line" >  
 <h2>柱形图-2 </h2>  
 <div class="chart">图表 </div>  
 <div class="panel-footer"></div>  
 </div>  
 <div class="panel pie" >  
 <h2>柱形图-3 </h2>  
 <div class="chart">图表 </div>  
 <div class="panel-footer"></div>  
 </div>  
 </div>  
   
  
 <div class="column">  
 <!-- no模块制作 -->  
 <div class="no">  
 <div class="no-hd"> <!-- ul>li\*2 tab键 -->  
 <ul>  
 <li>123456</li>  
 <li>111111</li>  
 </ul>  
 </div>  
 <div class="no-bd">  
 <ul>  
 <li>已关停网站数量</li>  
 <li>下架侵权作品数量</li>  
 </ul>  
 </div>  
 </div>  
 <div class="map">  
 <!-- 地图模块 -->  
 <div class="map1"></div>  
 <div class="map2"></div>  
 <div class="map3"></div>  
 <div class="chart">地图</div>  
 </div>  
 </div>  
  
 <div class="column">  
 <div class="panel bar2" >  
 <h2>柱形图-4 </h2>  
 <div class="chart">图表 </div>  
 <div class="panel-footer"></div>  
 </div>  
 <div class="panel line2" >  
 <h2>柱形图-5 </h2>  
 <div class="chart">图表 </div>  
 <div class="panel-footer"></div>  
 </div>   
 <div class="panel pie2" >  
 <h2>柱形图-6 </h2>  
 <div class="chart">图表 </div>  
 <div class="panel-footer"></div>  
 </div>  
 </div>  
 </section>  
 <script src="js/flexible.js"></script>  
  
 <script src="js/echarts.min.js"></script>  
 <script src="js/sample\_index.js"></script>  
</body>  
  
</html>

# confirmation\_index.js

update();  
***self***.setInterval("update()", 10000); //毫秒  
  
function update() {  
 //confirmation  
 //右上（当前不同作品类型存证数量分布 饼图）  
 (function() {  
  
 var flagRectrive = false;  
  
 $.ajax({  
 url: "/backend/certificateAmountGroupByWorkType",  
 type: "get",  
 dataType: "json",  
 success: function(response) {  
 flagRectrive = true; // 收到有效数据  
 ***console***.log('右上success')  
 var num = [];  
  
 response.data.forEach(function(value, index, array) {  
 var obj = new ***Object***();  
 obj.name = array[index]['workType'];  
 obj.value = array[index]['CertificateAmount'];  
 num.push(obj)  
 });  
  
 draw(num);  
 },  
 error: function() {  
 ***console***.log('右上error');  
  
 var num = [];  
 var obj0 = { name: "文字", value: 1886 };  
 var obj1 = { name: "音乐", value: 1758 };  
 var obj2 = { name: "图形", value: 1793 }  
  
 num.push(obj0);  
 num.push(obj1);  
 num.push(obj2);  
  
 // 还未收到有效数据  
 if (flagRectrive == false) {  
 draw(num);  
 }  
 }  
 });  
  
 function draw(num) {  
 //1. 实例化对象  
 var myChart = echarts.init(***document***.querySelector(".upperright .chart"));  
 //2. 指定配置和数据  
  
 var option = {  
  
 color: ['#60cda0', '#ed8884', '#ff9f7f', '#0096ff', '#9fe6b8', '#32c5e9', '#1d9dff'],  
  
 legend: {  
 left: 'center',  
 top: 'bottom',  
 bottom: "0",  
 itemHeight: 10,  
 itemWidth: 10,  
 textStyle: {  
 color: "rgba(255,255,255,.5)",  
 fontSize: "8"  
 }  
 },  
 tooltip: {  
 trigger: 'item',  
 formatter: '{a} <br/>{b} : {c} ({d}%)',  
 textStyle: {  
 //color: "rgba(255,255,255,.5)",  
 fontSize: "8"  
 }  
 },  
 series: [{  
 name: '存证数量',  
 type: 'pie',  
 radius: ['50%', '75%'],  
 avoidLabelOverlap: true,  
 itemStyle: {  
 borderRadius: 10,  
 borderColor: '#fff',  
 borderWidth: 2  
 },  
 label: {  
 show: false,  
 position: 'center'  
 },  
 emphasis: {  
 label: {  
 show: true,  
 fontSize: '10',  
 fontWeight: 'bold'  
 }  
 },  
 labelLine: {  
 show: false  
 },  
 data: num  
 }]  
 };  
  
 //option['series']['data'].push(value: CertificateAmount[0], n  
  
 //3. 把配置给实例对象  
 myChart.setOption(option);  
 //4. 图表适应屏幕  
 ***window***.addEventListener("resize", function() {  
 myChart.resize();  
 });  
  
 }  
  
 })();  
  
 //右中（个人账户与非个人账户接收者通证数量对比）  
 (function() {  
  
 var flagRectrive = false;  
  
 $.ajax({  
 url: "/backend/copyRightAmountGroupByIDtype",  
 type: "get",  
 dataType: "json",  
 success: function(response) {  
 flagRectrive = true; // 收到有效数据  
 ***console***.log('右中success')  
  
 var num = [];  
 var KEY = ***Object***.keys(response.data);  
 var obj0 = { value: response.data[KEY[0]], name: KEY[0] };  
 var obj1 = { value: response.data[KEY[1]], name: KEY[1] };  
 num.push(obj0);  
 num.push(obj1);  
 draw(num);  
 },  
 error: function() {  
 ***console***.log('右中error');  
  
 var num = [];  
 var obj0 = { value: 18950, name: "非个人账户" };  
 var obj1 = { value: 28881, name: "个人账户" };  
  
 num.push(obj0);  
 num.push(obj1);  
  
 // 还未收到有效数据  
 if (flagRectrive == false) {  
 draw(num);  
 }  
 }  
 });  
  
 function draw(num) {  
 //1. 实例d化对象  
 var myChart = echarts.init(***document***.querySelector(".middleright .chart"));  
 //2. 指定配置和数据  
  
 var option = {  
 color: ['#9fe6b8', '#1d9dff'],  
 title: {  
 left: 'center'  
 },  
 tooltip: {  
 trigger: 'item',  
 formatter: '{a} <br/>{b} : {c} ({d}%)',  
 textStyle: {  
 //color: "rgba(255,255,255,.5)",  
 fontSize: "8"  
 }  
 },  
 legend: {  
 left: 'center',  
 top: 'bottom',  
 bottom: "0",  
 itemHeight: 10,  
 itemWidth: 10,  
 textStyle: {  
 color: "rgba(255,255,255,.5)",  
 fontSize: "8"  
 }  
 },  
  
 series: [{  
 name: '通证接受者',  
 type: 'pie',  
 radius: '65%',  
 data: num,  
 emphasis: {  
 itemStyle: {  
 shadowBlur: 10,  
 shadowOffsetX: 0,  
 shadowColor: 'rgba(0, 0, 0, 0.5)'  
 }  
 }  
 }]  
 };  
  
 //3. 把配置给实例对象  
 myChart.setOption(option);  
 //4. 图表适应屏幕  
 ***window***.addEventListener("resize", function() {  
 myChart.resize();  
 });  
 }  
  
 })();  
  
 //右下（不同类别通证的数量分布 饼图）  
 (function() {  
  
 var flagRectrive = false;  
  
 $.ajax({  
 url: "/backend/copyRightAmountGroupByCopyrightType",  
 type: "get",  
 dataType: "json",  
 success: function(response) {  
 flagRectrive = true; // 收到有效数据  
 ***console***.log('右下success')  
  
 var num = [];  
 var KEY = ***Object***.keys(response.data);  
 for (var i = 0; i < KEY.length; i++) {  
 var obj = new ***Object***();  
 obj.value = response.data[KEY[i]]  
 obj.name = KEY[i]  
 num.push(obj)  
 }  
  
 draw(num);  
 },  
 error: function() {  
 ***console***.log('右下error');  
  
 var num = [];  
 num = [{ value: 10981, name: "复制权" }, { value: 4335, name: "发行权" },  
 { value: 4335, name: "出租权" }, { value: 4335, name: "展览权" },  
 { value: 4335, name: "表演权" }, { value: 4335, name: "放映权" }  
 ]  
  
 // 还未收到有效数据  
 if (flagRectrive == false) {  
 draw(num);  
 }  
 }  
 });  
  
 function draw(num) {  
 //1. 实例化对象  
 var myChart = echarts.init(***document***.querySelector(".lowerright .chart"));  
 //2. 指定配置和数据  
 var option = {  
  
 color: ['#006cff', '#60cda0', '#ed8884', '#ff9f7f', '#0096ff', '#9fe6b8', '#32c5e9', '#1d9dff'],  
  
 tooltip: {  
 trigger: 'item',  
 formatter: '{a} <br/>{b} : {c} ({d}%)',  
 textStyle: {  
 //color: "rgba(255,255,255,.5)",  
 fontSize: "8"  
 }  
 },  
 legend: {  
 left: 'center',  
 top: 'bottom',  
 bottom: "0",  
 itemHeight: 10,  
 itemWidth: 10,  
 textStyle: {  
 color: "rgba(255,255,255,.5)",  
 fontSize: "8"  
 }  
 },  
  
 series: [{  
 name: '通证数量',  
 type: 'pie',  
 top: "0%",  
 radius: ['10%', '60%'],  
 center: ["50%", "50%"],  
 roseType: "area",  
 // 文本标签控制饼形图文字的相关样式， 注意它是一个对象  
 label: {  
 fontSize: 13  
 },  
 //改变两条线  
 labelLine: {  
 length: 2,  
 length2: 8  
 },  
 itemStyle: {  
 borderRadius: 5  
 },  
 //存证、确权  
 //只截取所有取值的前六项  
 data: num.slice(1, 6)  
 }]  
 };  
 //3. 把配置给实例对象  
 myChart.setOption(option);  
 //4. 图表适应屏幕  
 ***window***.addEventListener("resize", function() {  
 myChart.resize();  
 });  
 }  
  
 })();  
  
 //左下（存证总数量随时间变化 折线图）  
 (function() {  
  
 var flagRectrive = false;  
  
 $.ajax({  
 url: "/backend/certificateAmountEXchange",  
 type: "get",  
 dataType: "json",  
 success: function(response) {  
 flagRectrive = true; // 收到有效数据  
 ***console***.log('左下success');  
  
 //两个数组  
 var num\_Month = [];  
 var num\_CertificateAmount = [];  
  
 response.data.forEach(function(value, index, array) {  
 num\_Month.push(array[index]['Month']);  
 num\_CertificateAmount.push(array[index]['CertificateAmount']);  
 });  
  
 draw(num\_Month, num\_CertificateAmount);  
 },  
 error: function() {  
 ***console***.log('左下error');  
  
 var num\_Month = ["七月", "八月", "九月", "十月", "十一月", "十二月", "一月", "二月", "三月", "四月", "五月", "六月"];  
 var num\_CertificateAmount = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7488, 3472]  
  
 // 还未收到有效数据  
 if (flagRectrive == false) {  
 draw(num\_Month, num\_CertificateAmount);  
 }  
 }  
 });  
  
 function draw(num\_Month, num\_CertificateAmount) {  
 //1. 实例化对象  
 var myChart = echarts.init(***document***.querySelector(".lowerleftleft .chart"));  
 //2. 指定配置和数据  
 var option = {  
  
 tooltip: {  
 trigger: 'axis',  
 },  
 legend: {  
 top: "0%",  
 textStyle: {  
 color: "rgba(255,255,255,.5)",  
 fontSize: "12"  
 }  
 },  
  
 grid: {  
 left: "10",  
 top: "30",  
 right: "10",  
 bottom: "10",  
 containLabel: true  
 },  
 xAxis: [{  
 // 文本颜色为rgba(255,255,255,.6) 文字大小为 12  
 axisLabel: {  
 textStyle: {  
 color: "rgba(255,255,255,.6)",  
 fontSize: 8  
 }  
 },  
 // x轴线的颜色为 rgba(255,255,255,.2)  
 axisLine: {  
 lineStyle: {  
 color: "rgba(255,255,255,.2)"  
 }  
 },  
 type: 'category',  
 boundaryGap: false,  
 // x轴更换数据  
 data: num\_Month,  
 }],  
 yAxis: [{  
 type: 'value',  
 min: 30, //设置y轴最小值  
 axisTick: {  
 show: false  
 },  
 axisLine: {  
 lineStyle: {  
 color: "rgba(255,255,255,.1)"  
 }  
 },  
 axisLabel: {  
 textStyle: {  
 color: "rgba(255,255,255,.6)",  
 fontSize: 12  
 }  
 },  
 // 修改分割线的颜色  
 splitLine: {  
 lineStyle: {  
 color: "rgba(255,255,255,.1)"  
 }  
 }  
  
 }],  
 series: [{  
 name: '',  
 type: 'line',  
 smooth: true,  
 //单独修改当前线条的样式  
 lineStyle: {  
 color: "#0184d5",  
 width: 2  
 },  
 //设置填充颜色  
 areaStyle: {  
 // 渐变色，只需要复制即可  
 color: new echarts.graphic.LinearGradient(  
 0,  
 0,  
 0,  
 1,  
 [{  
 offset: 0,  
 color: "rgba(1, 132, 213, 0.1)" // 渐变色的起始颜色  
 },  
 {  
 offset: 0.8, //偏移  
 color: "rgba(1, 132, 213, 0.1)" // 渐变线的结束颜色  
 }  
 ],  
 false  
 ),  
 shadowColor: "rgba(0, 0, 0, 0.1)"  
 },  
 emphasis: {  
 focus: 'series'  
 },  
 // 设置拐点 小圆点  
 symbol: "circle",  
 // 拐点大小  
 symbolSize: 8,  
 // 设置拐点颜色以及边框  
 itemStyle: {  
 color: "#0184d5",  
 borderColor: "rgba(221, 220, 107, .1)",  
 borderWidth: 12  
 },  
 // 开始不显示拐点， 鼠标经过显示  
 showSymbol: false,  
 // series 第一个对象data数据  
 data: num\_CertificateAmount,  
 },  
  
  
  
  
 {  
 name: "",  
 type: "line",  
 smooth: true,  
 lineStyle: {  
 normal: {  
 color: "#00d887",  
 width: 2  
 }  
 },  
 areaStyle: {  
 normal: {  
 color: new echarts.graphic.LinearGradient(  
 0,  
 0,  
 0,  
 1,  
 [{  
 offset: 0,  
 color: "rgba(0, 216, 135, 0.4)"  
 },  
 {  
 offset: 0.8,  
 color: "rgba(0, 216, 135, 0.1)"  
 }  
 ],  
 false  
 ),  
 shadowColor: "rgba(0, 0, 0, 0.1)"  
 }  
 },  
 // 设置拐点 小圆点  
 symbol: "circle",  
 // 拐点大小  
 symbolSize: 5,  
 // 设置拐点颜色以及边框  
 itemStyle: {  
 color: "#00d887",  
 borderColor: "rgba(221, 220, 107, .1)",  
 borderWidth: 12  
 },  
 // 开始不显示拐点， 鼠标经过显示  
 showSymbol: false,  
 // series 第二个对象data数据  
 //data: [130, 10, 20, 40, 30, 40, 80, 60, 20, 40, 90, 40, 20, 140, 30, 40, 130, 20, 20, 40, 80, 70, 30, 40, 30, 120, 20, 99, 50, 20]  
 },  
  
 ]  
 };  
 //3. 把配置给实例对象  
 myChart.setOption(option);  
 //4. 图表适应屏幕  
 ***window***.addEventListener("resize", function() {  
 myChart.resize();  
 });  
 }  
 })();  
  
 //中下（版权通证总数量随时间变化 折线图）  
 (function() {  
  
 var flagRectrive = false;  
  
 $.ajax({  
 url: "/backend/copyRightAmountEXchange",  
 type: "get",  
 dataType: "json",  
 success: function(response) {  
 flagRectrive = true; // 收到有效数据  
 ***console***.log('中下success');  
  
 var num\_Month = [];  
 var num\_CopyRightAmount = [];  
  
 response.data.forEach(function(value, index, array) {  
 num\_Month.push(array[index]['Month']);  
 num\_CopyRightAmount.push(array[index]['CopyRightAmount']);  
 });  
  
 draw(num\_Month, num\_CopyRightAmount);  
 },  
 error: function() {  
 ***console***.log('左下error');  
  
 var num\_Month = ["七月", "八月", "九月", "十月", "十一月", "十二月", "一月", "二月", "三月", "四月", "五月", "六月"];  
 var num\_CopyRightAmount = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 14905, 32950]  
  
 // 还未收到有效数据  
 if (flagRectrive == false) {  
 draw(num\_Month, num\_CopyRightAmount);  
 }  
 }  
 });  
  
 function draw(num\_Month, num\_CopyRightAmount) {  
 //1. 实例化对象  
 var myChart = echarts.init(***document***.querySelector(".lowerleftright .chart"));  
 //2. 指定配置和数据  
 var option = {  
  
 tooltip: {  
 trigger: 'axis',  
 },  
 legend: {  
 top: "0%",  
 textStyle: {  
 color: "rgba(255,255,255,.5)",  
 fontSize: "12"  
 }  
 },  
  
 grid: {  
 left: "10",  
 top: "30",  
 right: "10",  
 bottom: "10",  
 containLabel: true  
 },  
 xAxis: [{  
 // 文本颜色为rgba(255,255,255,.6) 文字大小为 12  
 axisLabel: {  
 textStyle: {  
 color: "rgba(255,255,255,.6)",  
 fontSize: 8  
 }  
 },  
 // x轴线的颜色为 rgba(255,255,255,.2)  
 axisLine: {  
 lineStyle: {  
 color: "rgba(255,255,255,.2)"  
 }  
 },  
 type: 'category',  
 boundaryGap: false,  
 // x轴更换数据  
 data: num\_Month,  
 }],  
 yAxis: [{  
 min: 30,  
 type: 'value',  
 axisTick: {  
 show: false  
 },  
 axisLine: {  
 lineStyle: {  
 color: "rgba(255,255,255,.1)"  
 }  
 },  
 axisLabel: {  
 textStyle: {  
 color: "rgba(255,255,255,.6)",  
 fontSize: 12  
 }  
 },  
 // 修改分割线的颜色  
 splitLine: {  
 lineStyle: {  
 color: "rgba(255,255,255,.1)"  
 }  
 }  
  
 }],  
 series: [{  
 name: '',  
 type: 'line',  
 smooth: true,  
 //单独修改当前线条的样式  
 lineStyle: {  
 color: "#0184d5",  
 width: 2  
 },  
 //设置填充颜色  
 areaStyle: {  
 // 渐变色，只需要复制即可  
 color: new echarts.graphic.LinearGradient(  
 0,  
 0,  
 0,  
 1,  
 [{  
 offset: 0,  
 color: "rgba(1, 132, 213, 0.1)" // 渐变色的起始颜色  
 },  
 {  
 offset: 0.8, //偏移  
 color: "rgba(1, 132, 213, 0.1)" // 渐变线的结束颜色  
 }  
 ],  
 false  
 ),  
 shadowColor: "rgba(0, 0, 0, 0.1)"  
 },  
 emphasis: {  
 focus: 'series'  
 },  
 // 设置拐点 小圆点  
 symbol: "circle",  
 // 拐点大小  
 symbolSize: 8,  
 // 设置拐点颜色以及边框  
 itemStyle: {  
 color: "#0184d5",  
 borderColor: "rgba(221, 220, 107, .1)",  
 borderWidth: 12  
 },  
 // 开始不显示拐点， 鼠标经过显示  
 showSymbol: false,  
 // series 第一个对象data数据  
 //data: [30, 40, 30, 40, 30, 40, 30, 60, 20, 40, 30, 40, 30, 40, 30, 40],  
 },  
  
  
  
  
 {  
 name: "",  
 type: "line",  
 smooth: true,  
 lineStyle: {  
 normal: {  
 color: "#00d887",  
 width: 2  
 }  
 },  
 areaStyle: {  
 normal: {  
 color: new echarts.graphic.LinearGradient(  
 0,  
 0,  
 0,  
 1,  
 [{  
 offset: 0,  
 color: "rgba(0, 216, 135, 0.4)"  
 },  
 {  
 offset: 0.8,  
 color: "rgba(0, 216, 135, 0.1)"  
 }  
 ],  
 false  
 ),  
 shadowColor: "rgba(0, 0, 0, 0.1)"  
 }  
 },  
 // 设置拐点 小圆点  
 symbol: "circle",  
 // 拐点大小  
 symbolSize: 5,  
 // 设置拐点颜色以及边框  
 itemStyle: {  
 color: "#00d887",  
 borderColor: "rgba(221, 220, 107, .1)",  
 borderWidth: 12  
 },  
 // 开始不显示拐点， 鼠标经过显示  
 showSymbol: false,  
 // series 第二个对象data数据  
 data: num\_CopyRightAmount  
 },  
  
 ]  
 };  
  
 //3. 把配置给实例对象  
 myChart.setOption(option);  
 //4. 图表适应屏幕  
 ***window***.addEventListener("resize", function() {  
 myChart.resize();  
 });  
 }  
 })();  
  
 //左上最大（不同作品类型的存证数量随时间的变化 柱形图）  
 (function() {  
  
 var flagRectrive = false;  
  
 $.ajax({  
 url: "/backend/certificateAmountGroupByWorkTypeEXchange",  
 type: "get",  
 dataType: "json",  
 success: function(response) {  
 flagRectrive = true; // 收到有效数据  
 ***console***.log('左上success');  
  
 // x轴月份  
 var x\_num = new ***Array***();  
 for (var i = 0; i < response.data.length; i++) {  
 x\_num.push(response.data[i][0]['Month'])  
 }  
  
 // 作品类型  
 var workType = new ***Array***();  
 for (var i = 0; i < response.data[0].length; i++) {  
 workType.push(response.data[0][i]['workType'])  
 }  
  
 // y轴数据  
 var y\_num = new ***Array***();  
  
 for (var i = 0; i < response.data[0].length; i++) { //4  
 var a = new ***Array***();  
 for (var j = 0; j < response.data.length; j++) { //3  
 a.push(response.data[j][i]['CertificateAmount']);  
 }  
 y\_num.push(a)  
 }  
 draw(x\_num, workType, y\_num)  
 },  
 error: function() {  
 ***console***.log('左下error');  
  
 var x\_num = ["七月", "十月", "一月", "四月"];  
 var workType = ["文字", "图形", "音乐"]  
 var y\_num = [  
 [0, 0, 0, 600],  
 [0, 0, 0, 590],  
 [0, 0, 0, 510]  
 ]  
  
 // 还未收到有效数据  
 if (flagRectrive == false) {  
 draw(x\_num, workType, y\_num)  
 }  
 }  
 });  
  
 function draw(x\_num, workType, y\_num) {  
 //1. 实例化对象  
 var myChart = echarts.init(***document***.querySelector(".upperleft .chart"));  
 //2. 指定配置和数据  
 var option = {  
  
 color: ['#87CEFA', '#6495ED', '#4169E1'],  
  
 legend: {  
 show: true,  
 },  
 grid: {  
 left: "0%",  
 top: "10%",  
 right: "0%",  
 bottom: "0%",  
 containLabel: true  
 },  
 tooltip: {  
 //show: false,  
 trigger: 'item',  
 //formatter: '{a} <br/>{b} : {c} ({d}%)',  
 textStyle: {  
 //color: "rgba(255,255,255,.5)",  
 fontSize: "8"  
 }  
 },  
 xAxis: {  
 type: 'category',  
 //data: ['二月', '五月', '八月', '十一月']  
 data: x\_num  
 },  
 yAxis: {  
 type: 'value'  
 },  
 series: [{  
 name: workType[0],  
 data: y\_num[0],  
 type: 'bar',  
 barWidth: 35,  
 showBackground: true,  
 backgroundStyle: {  
 color: 'rgba(180, 180, 180, 0.2)'  
 },  
 itemStyle: {  
 // 修改柱子圆角  
 barBorderRadius: 5  
 }  
 },  
 {  
 name: workType[1],  
 data: y\_num[1],  
 type: 'bar',  
 barWidth: 35,  
 showBackground: true,  
 backgroundStyle: {  
 color: 'rgba(180, 180, 180, 0.2)'  
 },  
 itemStyle: {  
 // 修改柱子圆角  
 barBorderRadius: 5  
 }  
 },  
 {  
 name: workType[2],  
 data: y\_num[2],  
 type: 'bar',  
 barWidth: 35,  
 showBackground: true,  
 backgroundStyle: {  
 color: 'rgba(180, 180, 180, 0.2)'  
 },  
 itemStyle: {  
 // 修改柱子圆角  
 barBorderRadius: 5  
 }  
 }  
 ]  
 };  
  
 //3. 把配置给实例对象  
 myChart.setOption(option);  
 //4. 图表适应屏幕  
 ***window***.addEventListener("resize", function() {  
 myChart.resize();  
 });  
 }  
  
 })();  
}

# surveillance\_index.js

//立即执行函数 mychart是局部变量；  
//多个立即执行函数中间有分号  
// (function(){  
// mychart =   
// })();  
  
update();  
***self***.setInterval("update()", 10000);  
  
function update() {  
  
 //左上 截止当前不同作品类型发生的侵权数量分布  
 (function() {  
  
 var flagRectrive = false;  
  
 $.ajax({  
 url: "/backend/listen/TortCountGroupByWorkType",  
 type: "get",  
 dataType: "json",  
 success: function(response) {  
 flagRectrive = true; // 收到有效数据  
 ***console***.log('左上success');  
  
 //两个数组  
 var workType = [];  
 var TortCount = [];  
  
 response.data.forEach(function(value, index, array) {  
 workType.push(array[index]['workType']);  
 TortCount.push(array[index]['TortCount']);  
 });  
  
 ***console***.log(workType)  
 ***console***.log(TortCount)  
 draw(workType, TortCount)  
 },  
 error: function() {  
 ***console***.log('左上error');  
 //两个数组  
 var workType = ["图形", "文字", "美术", "音乐", "口述", "杂技艺术"];  
 var TortCount = [185, 174, 159, 158, 57, 45];  
 draw(workType, TortCount)  
 }  
 });  
  
 function draw(workType, TortCount) {  
  
 //1. 实例化对象  
 var myChart = echarts.init(***document***.querySelector('.bar .chart'));  
 //2. 指定配置项和数据  
 var option = {  
 color: ["#2f89cf"],  
 tooltip: {  
 trigger: 'axis',  
 axisPointer: { // 坐标轴指示器，坐标轴触发有效  
 type: 'shadow' // 默认为直线，可选为：'line' | 'shadow'  
 }  
 },  
 grid: {  
 left: "0%",  
 top: "10px",  
 right: "3%",  
 bottom: "4%",  
 containLabel: true  
 },  
 xAxis: [{  
 type: 'category',  
 data: workType,  
 axisTick: {  
 alignWithLabel: true,  
 show: false,  
 interval: 1  
 },  
 //修改刻度标签  
 axisLabel: {  
 color: "rgba(255,255,255,.6)",  
 fontSize: "10"  
 },  
 axisLine: {  
 show: false  
 // 如果想要设置单独的线条样式   
 // lineStyle: {  
 // color: "rgba(255,255,255,.1)",  
 // width: 1,  
 // type: "solid"  
 }  
 }],  
 yAxis: [{  
 type: 'value',  
 // y 轴文字标签样式  
 axisLabel: {  
 color: "rgba(255,255,255,.6)",  
 fontSize: "12"  
 },  
 // y轴线条样式  
 axisLine: {  
 lineStyle: {  
 color: "rgba(255,255,255,.1)",  
 // width: 1,  
 // type: "solid"  
 }  
 },  
 // y 轴分隔线样式  
 splitLine: {  
 lineStyle: {  
 color: "rgba(255,255,255,.1)"  
 }  
 }  
  
 }],  
 series: [{  
 name: "侵权数量",  
 type: "bar",  
 // 修改柱子宽度  
 barWidth: "35%",  
  
 // series 更换数据 一般是通过ajax请求获取的  
 data: TortCount,  
 itemStyle: {  
 // 修改柱子圆角  
 barBorderRadius: 5  
 }  
 }]  
 };  
 //3. 把配置项给实例对象  
 myChart.setOption(option);  
 //4. 图表与屏幕自适应  
 ***window***.addEventListener("resize", function() {  
 myChart.resize();  
 });  
  
 }  
  
  
 })();  
  
 //柱状图2 右上 当前不同创作类型侵权数量  
 (function() {  
  
 var flagRectrive = false;  
  
 $.ajax({  
 url: "/backend/listen/TortCountGroupByCreationType",  
 type: "get",  
 dataType: "json",  
 success: function(response) {  
  
 flagRectrive = true; // 收到有效数据  
 ***console***.log('右上success');  
  
 //两个数组  
 var creationType = []; //1  
 var TortCount = []; //2  
 var TortCountPercentage = [];  
 var sum = 0;  
 var sumArr = new ***Array***(); //3  
  
 response.data.forEach(function(value, index, array) {  
 creationType.push(array[index]['creationType']);  
 TortCount.push(array[index]['TortCount']);  
 sum = sum + array[index]['TortCount'];  
 });  
  
 for (var i = 0; i < TortCount.length; i++) {  
 sumArr[i] = sum;  
 TortCountPercentage[i] = parseInt(100 \* TortCount[i] / sum);  
 }  
  
 ***console***.log(creationType)  
 ***console***.log(TortCount)  
 ***console***.log(sumArr)  
  
 draw(creationType, TortCount, sumArr);  
  
 },  
 error: function() {  
 ***console***.log('右上error');  
 var creationType = ["曲艺", "文字", "舞蹈", "杂技艺术", "口述", "音乐"]; //1  
 var TortCount = [360, 780, 156, 220, 390, 700]; //2  
 var sumArr = [940, 940, 940, 940, 940, 940]; //3  
 draw(creationType, TortCount, sumArr);  
 }  
 });  
  
 function draw(creationType, TortCount, sumArr) {  
  
 // 声明颜色数组  
 var myColor = ["#1089E7", "#F57474", "#56D0E3", "#F8B448", "#8B78F6"];  
  
 //1. 实例化对象  
 var myChart = echarts.init(***document***.querySelector('.bar2 .chart'));  
 //2. 指定配置和数据  
 var option = {  
  
 legend: {  
 data: ['2011年', '2012年']  
 },  
 grid: {  
 top: '10%',  
 left: '22%',  
 right: '22%',  
 bottom: '10%',  
 // containLabel: true  
 },  
 xAxis: {  
 show: false  
 },  
 yAxis: [  
 // Y轴的第一组对象，左边的文字  
 {  
 //不显示y轴线条  
 axisLine: {  
 show: false  
 },  
 // 不显示刻度  
 axisTick: {  
 show: false  
 },  
 //y轴文字设置为白色  
 axisLabel: {  
 color: "#fff"  
 },  
 type: 'category',  
 inverse: true,  
 data: creationType,  
 },  
 // Y轴的第二组对象，右边的文字  
 {  
 show: true,  
 inverse: true,  
 data: sumArr,  
 // 不显示y轴的线  
 axisLine: {  
 show: false  
 },  
 // 不显示刻度  
 axisTick: {  
 show: false  
 },  
 axisLabel: {  
 textStyle: {  
 fontSize: 12,  
 color: "#fff"  
 }  
 }  
 },  
 ],  
 series: [{  
 type: 'bar',  
 data: TortCount,  
 name: "条",  
 // 柱子之间的距离  
 barCategoryGap: 50,  
 //柱子的宽度  
 barWidth: 10,  
 yAxisIndex: 0,  
 // 柱子设为圆角  
 itemStyle: {  
 normal: {  
 barBorderRadius: 20,  
 // 此时的color可以修改柱子的颜色  
 // params是六个柱子的对象 dataIndex是当前柱子索引号  
 color: function name(params) {  
 return myColor[params.dataIndex];  
 }  
 }  
 },  
 // 图形上的文本标签  
 label: {  
 normal: {  
 show: true,  
 // 图形内显示  
 position: "inside",  
 // 文字的显示格式  
 formatter: "{c}"  
 }  
 }  
 },  
  
 {  
 name: "框",  
 type: 'bar',  
 barCategoryGap: 50,  
 barWidth: 15,  
 yAxisIndex: 1,  
 itemStyle: {  
 color: "none",  
 borderColor: "#00c1de",  
 borderWidth: 3,  
 barBorderRadius: 15  
 },  
 data: sumArr  
 }  
 ]  
 };  
 //3. 把配置给实例对象  
 myChart.setOption(option);  
 //4. 图表与屏幕自适应  
 ***window***.addEventListener("resize", function() {  
 myChart.resize();  
 });  
 }  
  
 })();  
  
 //折线图1 左中 不同作品类型的侵权数量变化  
 (function() {  
  
 var flagRectrive = false;  
  
 $.ajax({  
 url: "/backend/listen/TortCountGroupByWorkTypeEXchange",  
 type: "get",  
 dataType: "json",  
 success: function(response) {  
  
 flagRectrive = true; // 收到有效数据  
 ***console***.log('左中success');  
  
 // x轴月份  
 var x\_num = new ***Array***(); //1  
 for (var i = 0; i < response.data.length; i++) {  
 x\_num.push(response.data[i][0]['Month'])  
 }  
 ***console***.log(x\_num);  
  
  
 // 作品类型 ["文字","口述","音乐"]  
 var workType = new ***Array***(); //2  
 for (var i = 0; i < response.data[0].length; i++) {  
 workType.push(response.data[0][i]['workType'])  
 }  
 ***console***.log(workType);  
  
 // y轴数据  
 var y\_num = new ***Array***(); //3  
 for (var i = 0; i < response.data[0].length; i++) { //4  
 var a = new ***Array***();  
 for (var j = 0; j < response.data.length; j++) { //3  
 a.push(response.data[j][i]['TortCount']);  
 }  
 y\_num.push(a)  
 }  
 ***console***.log(y\_num)  
  
 draw(x\_num, workType, y\_num);  
  
 },  
 error: function() {  
 ***console***.log('左中error');  
 var x\_num = ["八月", "九月", "十月", "十一月", "十二月", "一月", "二月", "三月", "四月", "五月", "六月", "七月"]; //1  
 var workType = ["图形", "文字", "美术"]; //2  
 var y\_num = [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 185, 0],[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 174, 0],[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 159, 0]]; //3  
 draw(x\_num, workType, y\_num);  
 }  
 });  
  
 function draw(x\_num, workType, y\_num) {  
  
 //1. 实例化对象  
 var myChart = echarts.init(***document***.querySelector(".line .chart"));  
 //2. 指定配置和数据  
 var option = {  
  
 tooltip: {  
 trigger: 'axis'  
 },  
 // 图例组件  
 legend: {  
 textStyle: {  
 color: '#4c9bfd' // 图例文字颜色  
 },  
 right: '10%' // 距离右边10%  
 },  
 grid: {  
 top: '20%',  
 left: '3%',  
 right: '4%',  
 bottom: '3%',  
 show: true, // 显示边框  
 borderColor: '#012f4a', // 边框颜色  
 containLabel: true // 包含刻度文字在内  
 },  
 xAxis: {  
 type: 'category',  
 data: x\_num,  
 axisTick: {  
 show: false // 去除刻度线  
 },  
 axisLabel: {  
 color: '#4c9bfd' // 文本颜色  
 },  
 axisLine: {  
 show: false // 去除轴线  
 },  
 boundaryGap: false // 去除轴内间距  
 },  
 yAxis: {  
 type: 'value',  
 axisTick: {  
 show: false // 去除刻度  
 },  
 axisLabel: {  
 color: '#4c9bfd' // 文字颜色  
 },  
 splitLine: {  
 lineStyle: {  
 color: '#012f4a' // 分割线颜色  
 }  
 }  
 },  
 color: ['#00f2f1', '#ed3f35', '#FFFF00'],  
 series: [{  
 name: workType[0],  
 data: y\_num[0],  
 type: 'line',  
 smooth: true  
 }, {  
 name: workType[1],  
 data: y\_num[1],  
 type: 'line',  
 smooth: true  
  
 },  
 {  
 name: workType[2],  
 data: y\_num[2],  
 type: 'line',  
 smooth: true  
  
 }  
 ]  
 };  
 //3. 把配置给实例对象  
 myChart.setOption(option);  
 //4. 图表与屏幕自适应  
 ***window***.addEventListener("resize", function() {  
 myChart.resize();  
 });  
  
 }  
  
 })();  
  
 //折线图2 右中 不同创作类型的侵权数量变化  
 (function() {  
  
 var flagRectrive = false;  
  
 $.ajax({  
 url: "/backend/listen/TortCountGroupByCreationTypeEXchange",  
 type: "get",  
 dataType: "json",  
 success: function(response) {  
  
 flagRectrive = true; // 收到有效数据  
 ***console***.log('右中success');  
  
 // x轴月份  
 var x\_num = new ***Array***();//1  
 for (var i = 0; i < response.data.length; i++) {  
 x\_num.push(response.data[i][0]['Month'])  
 }  
 ***console***.log(x\_num);  
  
  
 // 作品类型 ["文字","口述","音乐"]  
 var creationType = new ***Array***();//2  
 for (var i = 0; i < response.data[0].length; i++) {  
 creationType.push(response.data[0][i]['creationType'])  
 }  
 ***console***.log(creationType);  
  
 // y轴数据  
 var y\_num = new ***Array***();//3  
 for (var i = 0; i < response.data[0].length; i++) { //4  
 var a = new ***Array***();  
 for (var j = 0; j < response.data.length; j++) { //3  
 a.push(response.data[j][i]['TortCount']);  
 }  
 y\_num.push(a)  
 }  
 ***console***.log(y\_num)  
  
 draw(x\_num,creationType,y\_num)  
   
 },  
 error: function() {  
 ***console***.log('右中error');  
 var x\_num = ["八月", "九月", "十月", "十一月", "十二月", "一月", "二月", "三月", "四月", "五月", "六月", "七月"];//1  
 var creationType = ["曲艺", "文字", "舞蹈"];//2  
 var y\_num = [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 166, 0],[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 164, 0],[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 156, 0]]; //3  
 draw(x\_num,creationType,y\_num)  
 }  
 });  
  
 function draw(x\_num,creationType,y\_num){  
  
 //1. 实例化对象  
 var myChart = echarts.init(***document***.querySelector(".line2 .chart"));  
 //2. 指定配置和数据  
 var option = {  
  
 tooltip: {  
 trigger: 'axis',  
 },  
 legend: {  
 top: "0%",  
 textStyle: {  
 color: "rgba(255,255,255,.5)",  
 fontSize: "12"  
 }  
 },  
  
 grid: {  
 left: "10",  
 top: "30",  
 right: "10",  
 bottom: "10",  
 containLabel: true  
 },  
 xAxis: [{  
 // 文本颜色为rgba(255,255,255,.6) 文字大小为 12  
 axisLabel: {  
 textStyle: {  
 color: "rgba(255,255,255,.6)",  
 fontSize: 12  
 }  
 },  
 // x轴线的颜色为 rgba(255,255,255,.2)  
 axisLine: {  
 lineStyle: {  
 color: "rgba(255,255,255,.2)"  
 }  
 },  
 type: 'category',  
 boundaryGap: false,  
 // x轴更换数据  
 data: x\_num,  
 }],  
 yAxis: [{  
 type: 'value',  
 axisTick: {  
 show: false  
 },  
 axisLine: {  
 lineStyle: {  
 color: "rgba(255,255,255,.1)"  
 }  
 },  
 axisLabel: {  
 textStyle: {  
 color: "rgba(255,255,255,.6)",  
 fontSize: 12  
 }  
 },  
 // 修改分割线的颜色  
 splitLine: {  
 lineStyle: {  
 color: "rgba(255,255,255,.1)"  
 }  
 }  
  
 }],  
 series: [{  
 name: creationType[0],  
 type: 'line',  
 smooth: true,  
 //单独修改当前线条的样式  
 lineStyle: {  
 color: "#0184d5",  
 width: 2  
 },  
 //设置填充颜色  
 areaStyle: {  
 // 渐变色，只需要复制即可  
 color: new echarts.graphic.LinearGradient(0, 0, 0, 1,  
 [{  
 offset: 0,  
 color: "rgba(1, 132, 213, 0.1)" // 渐变色的起始颜色  
 },  
 {  
 offset: 0.8, //偏移  
 color: "rgba(1, 132, 213, 0.1)" // 渐变线的结束颜色  
 }  
 ],  
 false  
 ),  
 shadowColor: "rgba(0, 0, 0, 0.1)"  
 },  
 emphasis: {  
 focus: 'series'  
 },  
 // 设置拐点 小圆点  
 symbol: "circle",  
 // 拐点大小  
 symbolSize: 8,  
 // 设置拐点颜色以及边框  
 itemStyle: {  
 color: "#0184d5",  
 borderColor: "rgba(221, 220, 107, .1)",  
 borderWidth: 12  
 },  
 // 开始不显示拐点， 鼠标经过显示  
 showSymbol: false,  
 // series 第一个对象data数据  
 data: y\_num[0]  
 },  
  
  
 {  
 name: creationType[1],  
 type: 'line',  
 smooth: true,  
 lineStyle: {  
 normal: {  
 color: "#00d887",  
 width: 2  
 }  
 },  
 areaStyle: {  
 normal: {  
 color: new echarts.graphic.LinearGradient(0, 0, 0, 1,  
 [{  
 offset: 0,  
 color: "rgba(0, 216, 135, 0.4)"  
 },  
 {  
 offset: 0.8,  
 color: "rgba(0, 216, 135, 0.1)"  
 }  
 ],  
 false  
 ),  
 shadowColor: "rgba(0, 0, 0, 0.1)"  
 }  
 },  
 // 设置拐点 小圆点  
 symbol: "circle",  
 // 拐点大小  
 symbolSize: 5,  
 // 设置拐点颜色以及边框  
 itemStyle: {  
 color: "#00d887",  
 borderColor: "rgba(221, 220, 107, .1)",  
 borderWidth: 12  
 },  
 // 开始不显示拐点， 鼠标经过显示  
 showSymbol: false,  
 // series 第二个对象data数据  
 data: y\_num[1]  
 },  
  
 {  
 name: creationType[2],  
 type: 'line',  
 smooth: true,  
 lineStyle: {  
 normal: {  
 color: "#DCDCDC",  
 width: 2  
 }  
 },  
 areaStyle: {  
 normal: {  
 color: new echarts.graphic.LinearGradient(0, 0, 0, 1,  
 [{  
 offset: 0,  
 color: "rgba(245, 245, 245, 0.4)"  
 },  
 {  
 offset: 0.8,  
 color: "rgba(245, 245, 245, 0.1)"  
 }  
 ],  
 false  
 ),  
 shadowColor: "rgba(0, 0, 0, 0.1)"  
 }  
 },  
 // 设置拐点 小圆点  
 symbol: "circle",  
 // 拐点大小  
 symbolSize: 5,  
 // 设置拐点颜色以及边框  
 itemStyle: {  
 color: "#DCDCDC",  
 borderColor: "rgba(245, 245, 245, .1)",  
 borderWidth: 12  
 },  
 // 开始不显示拐点， 鼠标经过显示  
 showSymbol: false,  
 // series 第二个对象data数据  
 data: y\_num[2]  
 },  
  
  
 ]  
 };  
  
 //3. 把配置给实例对象  
 myChart.setOption(option);  
 //4. 图表与屏幕自适应  
 ***window***.addEventListener("resize", function() {  
 myChart.resize();  
 });  
  
 }  
  
 })();  
  
 //饼形图1 左下 侵权数量前三的站点  
 (function test() {  
  
 var flagRectrive = false;  
  
 $.ajax({  
 url: "/backend/listen/TortCountGroupByTortSite",  
 type: "get",  
 dataType: "json",  
 success: function(response) {  
  
 flagRectrive = true; // 收到有效数据  
 ***console***.log('左下success');  
  
 var num = [];  
 response.data.forEach(function(value, index, array) {  
 var obj = new ***Object***();  
 obj.name = array[index]['TortSite'];  
 obj.value = array[index]['TortCount'];  
 num.push(obj)  
 });  
 ***console***.log(num);  
  
 draw(num);  
  
 },  
 error: function() {  
 ***console***.log('左下error');  
 var num = [{name: "乐视", value: 577}, {name: "新浪", value: 233}, {name: "酷狗", value: 187}];  
 draw(num);  
 }  
 });  
   
 function draw(num){  
  
 //1. 实例化对象  
 var myChart = echarts.init(***document***.querySelector(".pie .chart"));  
 //2. 指定配置和数据  
 var option = {  
 color: [  
 "#065aab",  
 "#066eab",  
 "#0682ab",  
 "#0696ab",  
 "#06a0ab",  
 ],  
 grid: {  
 left: "10",  
 top: "30",  
 right: "10",  
 bottom: "10",  
 containLabel: true  
 },  
 tooltip: {  
 trigger: 'item'  
 },  
 legend: {  
 // 距离底部为0%  
 bottom: "0%",  
 // 小图标的宽度和高度  
 itemWidth: 10,  
 itemHeight: 10,  
 // 修改图例组件的文字为 12px  
 textStyle: {  
 color: "rgba(255,255,255,.5)",  
 fontSize: "9"  
 },  
 // legend 中的data 可省略  
 //data: ["视频类平台", "社交网络", "搜索引擎", "小网站", "小网站"],  
 },  
  
 series: [{  
 avoidLabelOverlap: false,  
 name: "侵权数量",  
 type: "pie",  
 // 设置饼形图在容器中的位置  
 center: ["50%", "43%"],  
 // 修改内圆半径和外圆半径为 百分比是相对于容器宽度来说的  
 radius: ["40%", "60%"],  
 // 不显示标签文字  
 label: {  
 show: false  
 },  
 // 不显示连接线  
 labelLine: {  
 show: false  
 },  
  
 // series 中的数据  
 data: num,  
 }]  
 };  
 //3. 把配置给实例对象  
 myChart.setOption(option);  
 //4. 图表与屏幕自适应  
 ***window***.addEventListener("resize", function() {  
 myChart.resize();  
 });  
  
 }  
  
 })();  
  
 //饼形图2 右下   
 (function() {  
  
 var flagRectrive = false;  
  
 $.ajax({  
 url: "/backend/listen/TortCountGroupByTortSiteGroupByWorkType",  
 type: "get",  
 dataType: "json",  
 success: function(response) {  
  
 flagRectrive = true; // 收到有效数据  
 ***console***.log('右下success');  
  
 var num = [];  
 response.data.forEach(function(value, index, array) {  
 var obj = new ***Object***();  
 obj.name = array[index]['workType'];  
 obj.value = array[index]['TortCount'];  
 num.push(obj)  
 });  
 ***console***.log(num);  
  
 draw(num);  
 },  
 error: function() {  
 ***console***.log('右下error');  
 var num = [{name: "图形", value: 176}, {name: "文字", value: 160}, {name: "美术", value: 147}, {name: "音乐", value: 142}, {name: "口述", value: 56}, {name: "曲艺", value: 43}]  
 draw(num);  
 }  
 });  
  
 function draw(num){  
  
 //1. 实例化对象   
 var myChart = echarts.init(***document***.querySelector(".pie2 .chart"));  
  
 //2. 指定配置和数据  
 var option = {  
  
 color: ['#006cff', '#60cda0', '#ed8884', '#ff9f7f', '#0096ff', '#9fe6b8', '#32c5e9', '#1d9dff'],  
  
 tooltip: {  
 trigger: 'item',  
 formatter: '{a} <br/>{b} : {c} ({d}%)'  
 },  
 legend: {  
 left: 'center',  
 top: 'bottom',  
 bottom: "0",  
 itemHeight: 5,  
 itemWidth: 5,  
 textStyle: {  
 color: "rgba(255,255,255,.5)",  
 fontSize: "8"  
 }  
 },  
  
 series: [{  
 name: '被侵权作品数量',  
 type: 'pie',  
 top: "0%",  
 radius: ['10%', '60%'],  
 center: ["50%", "50%"],  
 roseType: "area",  
 // 文本标签控制饼形图文字的相关样式， 注意它是一个对象  
 label: {  
 fontSize: 10  
 },  
 //改变两条线  
 labelLine: {  
 length: 2,  
 length2: 8  
 },  
 itemStyle: {  
 borderRadius: 5  
 },  
 data: num  
 }]  
 };  
 //3. 把配置给实力对象  
 myChart.setOption(option);  
 //4. 图表适应屏幕  
 ***window***.addEventListener("resize", function() {  
 myChart.resize();  
 });  
  
 }  
  
  
 })();  
  
 //中间部分  
 (function() {  
  
 var flagRectrive = false;  
  
 $.ajax({  
 url: "/backend/listen/Tort\_AND\_ClaimCountGroupByWorkType",  
 type: "get",  
 dataType: "json",  
 success: function(response) {  
  
 flagRectrive = true; // 收到有效数据  
 ***console***.log('中间success');  
  
 var num = [];  
 var ClaimCount = [];  
 var TortCount = [];  
  
 response.data.forEach(function(value, index, array) {  
 var obj = new ***Object***();  
 obj.name = array[index]['workType'];  
 obj.max = array[index]['TortCount'];  
 ClaimCount.push(array[index]['ClaimCount'])  
 TortCount.push(array[index]['TortCount'])  
 num.push(obj)  
 });  
  
 ***console***.log(ClaimCount);  
 // console.log(TortCount);  
 ***console***.log(num);  
  
 draw(num,ClaimCount)  
   
 },  
 error: function() {  
 ***console***.log('中间error');  
  
 var ClaimCount = [80, 100, 20, 70, 10, 3];  
 var num = [{name: "图形", max: 185}, {name: "文字", max: 174}, {name: "美术", max: 159}, {name: "音乐", max: 158}, {name: "口述", max: 57}, {name: "曲艺", max: 45}];  
  
 draw(num,ClaimCount);  
 }  
 });  
  
 function draw(num,ClaimCount){  
  
 //1. 实例化对象  
 var myChart = echarts.init(***document***.querySelector(".map .chart"));  
 //2. 指定配置和数据  
 var option = {  
 color: ['#00FFFF'],  
 // '#9fe6b8','#32c5e9',   
  
 title: {  
 text: '侵权数量前三站点发起维权作品类型分布',  
 left: 'center',  
 textStyle: {  
 color: '#fff',  
 fontSize: 15  
 },  
 padding: 20,  
 fontWeight: 'normal',  
 },  
  
 tooltip: {},  
 legend: {  
 // data: ['维权类型数量'],  
 textStyle: {  
 color: "rgba(255,255,255,.5)",  
 fontSize: "15",  
 },  
 top: 'bottom'  
 },  
 radar: {  
 // shape: 'circle',  
 radius: "70%",  
 name: {  
 textStyle: {  
 color: '#fff',  
 //backgroundColor: '#999',  
 borderRadius: 10,  
 padding: [3, 10]  
 }  
 },  
 indicator: num  
 },  
 series: [{  
 type: 'radar',  
 // areaStyle: {normal: {}},  
 data: [{  
 value: ClaimCount,  
 name: '发起维权作品数量'  
 }]  
 }]  
 };  
  
 //3. 把配置给实例对象  
 myChart.setOption(option);  
 //4. 图表与屏幕自适应  
 ***window***.addEventListener("resize", function() {  
 myChart.resize();  
 });  
  
 }  
  
  
 })();  
  
  
  
}

# confirmation.css

\* {  
 margin: 0;  
 padding: 0;  
 box-sizing: border-box;  
}  
body {  
 background: url(../images/bg.jpg) no-repeat top center;  
 line-height: 1.15;  
}  
header {  
 position: relative;  
 height: 1.25rem;  
 background: url(../images/head\_bg.png) no-repeat;  
 background-size: 100% 100%;  
}  
header h1 {  
 font-size: 0.475rem;  
 color: #fff;  
 text-align: center;  
 line-height: 1rem;  
}  
header .showTime {  
 position: absolute;  
 top: 0;  
 right: 0.375rem;  
 line-height: 0.9375rem;  
 color: rgba(255, 255, 255, 0.7);  
 font-size: 0.25rem;  
}  
  
/\* Larger buttons styles \*/  
  
.button.large{  
 padding: 6px 15px;  
 text-transform: uppercase;  
}  
  
.button.large:active{  
 top: 2px;  
}  
  
/\* Colored buttons styles \*/  
  
.button.blue {  
 color: #fff;  
 text-shadow: 0 1px 0 rgba(0,0,0,.2);  
  
 background-image: -webkit-gradient(linear, left top, left bottom, from(rgba(255,255,255,.3)), to(rgba(255,255,255,0)));  
 background-image: -webkit-linear-gradient(top, rgba(255,255,255,.3), rgba(255,255,255,0));  
 background-image: -moz-linear-gradient(top, rgba(255,255,255,.3), rgba(255,255,255,0));  
 background-image: -ms-linear-gradient(top, rgba(255,255,255,.3), rgba(255,255,255,0));  
 background-image: -o-linear-gradient(top, rgba(255,255,255,.3), rgba(255,255,255,0));  
 background-image: linear-gradient(top, rgba(255,255,255,.3), rgba(255,255,255,0));  
}  
  
.button.blue{  
 background-color: #269CE9;  
 border-color: #269CE9;  
}  
  
.button.blue:hover{  
 background-color: #70B9E8;  
}  
  
.button.blue:active{  
 background: #269CE9;  
}  
  
.blue[disabled], .blue[disabled]:hover, .blue[disabled]:active{  
 border-color: #269CE9;  
 background: #269CE9;  
 color: #93D5FF;  
}  
  
.button.blueShadow {  
 color: #fff;  
 text-shadow: 0 1px 0 rgba(0,0,0,.2);  
  
 background-image: -webkit-gradient(linear, left top, left bottom, from(rgba(255,255,255,.3)), to(rgba(255,255,255,0)));  
 background-image: -webkit-linear-gradient(top, rgba(255,255,255,.3), rgba(255,255,255,0));  
 background-image: -moz-linear-gradient(top, rgba(255,255,255,.3), rgba(255,255,255,0));  
 background-image: -ms-linear-gradient(top, rgba(255,255,255,.3), rgba(255,255,255,0));  
 background-image: -o-linear-gradient(top, rgba(255,255,255,.3), rgba(255,255,255,0));  
 background-image: linear-gradient(top, rgba(255,255,255,.3), rgba(255,255,255,0));  
}  
  
.button.blueShadow{  
 background-color: White;  
 border-color: White;  
}  
  
.button.blueShadow:hover{  
 background-color: #70B9E8;  
}  
  
.button.blueShadow:active{  
 background: #93D5FF;  
}  
  
.blueShadow[disabled], .blueShadow[disabled]:hover, .blueShadow[disabled]:active{  
 border-color: #93D5FF;  
 background: #93D5FF;  
 color: #93D5FF;  
}  
  
/\* Group buttons \*/  
  
.button-group,  
.button-group li{  
 display: inline-block;  
 \*display: inline;  
 zoom: 1;  
 margin-right: -1px; /\* Overlap each right button border \*/  
}  
  
.button-group{  
 font-size: 0; /\* Inline block elements gap - fix \*/  
 margin: 0;  
 background: rgba(0, 0, 0, .1);  
 border-bottom: 3px rgba(0, 0, 0, .1);  
 padding: 3px;  
 -moz-border-radius: 3px;  
 -webkit-border-radius: 3px;  
 border-radius: 3px;  
}  
  
.button-group .button{  
 font-size: 13px; /\* Set the font size, different from inherited 0 \*/  
 -moz-border-radius: 0;  
 -webkit-border-radius: 0;  
 border-radius: 0;  
}  
  
.button-group .button:active{  
 -moz-box-shadow: 0 0 1px rgba(0, 0, 0, .2) inset, 5px 0 5px -3px rgba(0, 0, 0, .2) inset, -5px 0 5px -3px rgba(0, 0, 0, .2) inset;  
 -webkit-box-shadow: 0 0 1px rgba(0, 0, 0, .2) inset, 5px 0 5px -3px rgba(0, 0, 0, .2) inset, -5px 0 5px -3px rgba(0, 0, 0, .2) inset;  
 box-shadow: 0 0 1px rgba(0, 0, 0, .2) inset, 5px 0 5px -3px rgba(0, 0, 0, .2) inset, -5px 0 5px -3px rgba(0, 0, 0, .2) inset;  
}  
  
.button-group li:first-child .button{  
 -moz-border-radius: 3px 0 0 3px;  
 -webkit-border-radius: 3px 0 0 3px;  
 border-radius: 3px 0 0 3px;  
}  
  
.button-group li:first-child .button:active{  
 -moz-box-shadow: 0 0 1px rgba(0, 0, 0, .2) inset, -5px 0 5px -3px rgba(0, 0, 0, .2) inset;  
 -webkit-box-shadow: 0 0 1px rgba(0, 0, 0, .2) inset, -5px 0 5px -3px rgba(0, 0, 0, .2) inset;  
 box-shadow: 0 0 1px rgba(0, 0, 0, .2) inset, -5px 0 5px -3px rgba(0, 0, 0, .2) inset;  
}  
  
.button-group li:last-child .button{  
 -moz-border-radius: 0 3px 3px 0;  
 -webkit-border-radius: 0 3px 3px 0;  
 border-radius: 0 3px 3px 0;  
}  
  
.button-group li:last-child .button:active{  
 -moz-box-shadow: 0 0 1px rgba(0, 0, 0, .2) inset, 5px 0 5px -3px rgba(0, 0, 0, .2) inset;  
 -webkit-box-shadow: 0 0 1px rgba(0, 0, 0, .2) inset, 5px 0 5px -3px rgba(0, 0, 0, .2) inset;  
 box-shadow: 0 0 1px rgba(0, 0, 0, .2) inset, 5px 0 5px -3px rgba(0, 0, 0, .2) inset;  
}  
  
.mainbox {  
 display: flex;  
 min-width: 1024px;  
 max-width: 1920px;  
 margin: 0 auto;  
 padding: 0.125rem 0.125rem 0;  
}  
.mainbox .column {  
 flex: 8;  
}  
.mainbox .column .innerbox {  
 display: flex;  
 margin: 0 auto;  
}  
.mainbox .column .innerbox .innercolumn {  
 flex: 40;  
}  
.mainbox .column .innerbox .innercolumn:nth-child(2) {  
 flex: 1;  
 overflow: hidden;  
}  
.mainbox .column:nth-child(2) {  
 flex: 3;  
 margin: 0 0.125rem 0.1875rem;  
 overflow: hidden;  
}  
.mainbox .panel {  
 position: relative;  
 height: 7.954rem;  
 padding: 0 0.1875rem 0.5rem;  
 margin-bottom: 0.1875rem;  
 border: 1px solid rgba(25, 186, 139, 0.17);  
 background: url(../images/line.png) rgba(255, 255, 255, 0.04);  
}  
.mainbox .panel::before {  
 position: absolute;  
 top: 0;  
 left: 0;  
 width: 10px;  
 height: 10px;  
 border-left: 2px solid #02a6b5;  
 border-top: 2px solid #02a6b5;  
 content: "";  
}  
.mainbox .panel::after {  
 position: absolute;  
 top: 0;  
 right: 0;  
 width: 10px;  
 height: 10px;  
 border-right: 2px solid #02a6b5;  
 border-top: 2px solid #02a6b5;  
 content: "";  
}  
.mainbox .panel .panel-footer {  
 position: absolute;  
 left: 0;  
 bottom: 0;  
 width: 100%;  
}  
.mainbox .panel .panel-footer::before {  
 position: absolute;  
 bottom: 0;  
 left: 0;  
 content: "";  
 width: 10px;  
 height: 10px;  
 border-left: 2px solid #02a6b5;  
 border-bottom: 2px solid #02a6b5;  
}  
.mainbox .panel .panel-footer::after {  
 position: absolute;  
 bottom: 0;  
 right: 0;  
 content: "";  
 width: 10px;  
 height: 10px;  
 border-bottom: 2px solid #02a6b5;  
 border-right: 2px solid #02a6b5;  
}  
.mainbox .panel h2 {  
 height: 0.6rem;  
 color: #fff;  
 line-height: 0.6rem;  
 text-align: center;  
 font-size: 0.25rem;  
 font-weight: 400;  
}  
.mainbox .panel .chart {  
 height: 7rem;  
}  
.mainbox .innerpanel {  
 position: relative;  
 height: 3.875rem;  
 padding: 0 0.1875rem 0.5rem;  
 margin-bottom: 0.1875rem;  
 border: 1px solid rgba(25, 186, 139, 0.17);  
 background: url(../images/line.png) rgba(255, 255, 255, 0.04);  
}  
.mainbox .innerpanel::before {  
 position: absolute;  
 top: 0;  
 left: 0;  
 width: 10px;  
 height: 10px;  
 border-left: 2px solid #02a6b5;  
 border-top: 2px solid #02a6b5;  
 content: "";  
}  
.mainbox .innerpanel::after {  
 position: absolute;  
 top: 0;  
 right: 0;  
 width: 10px;  
 height: 10px;  
 border-right: 2px solid #02a6b5;  
 border-top: 2px solid #02a6b5;  
 content: "";  
}  
.mainbox .innerpanel .innerpanel-footer {  
 position: absolute;  
 left: 0;  
 bottom: 0;  
 width: 100%;  
}  
.mainbox .innerpanel .innerpanel-footer::before {  
 position: absolute;  
 bottom: 0;  
 left: 0;  
 content: "";  
 width: 10px;  
 height: 10px;  
 border-left: 2px solid #02a6b5;  
 border-bottom: 2px solid #02a6b5;  
}  
.mainbox .innerpanel .innerpanel-footer::after {  
 position: absolute;  
 bottom: 0;  
 right: 0;  
 content: "";  
 width: 10px;  
 height: 10px;  
 border-bottom: 2px solid #02a6b5;  
 border-right: 2px solid #02a6b5;  
}  
.mainbox .innerpanel h2 {  
 height: 0.6rem;  
 color: #fff;  
 line-height: 0.6rem;  
 text-align: center;  
 font-size: 0.25rem;  
 font-weight: 400;  
}  
.mainbox .innerpanel .chart {  
 height: 3rem;  
}  
.mainbox .rightpanel {  
 position: relative;  
 height: 3.875rem;  
 padding: 0 0.1875rem 0.5rem;  
 margin-bottom: 0.1875rem;  
 border: 1px solid rgba(25, 186, 139, 0.17);  
 background: url(../images/line.png) rgba(255, 255, 255, 0.04);  
}  
.mainbox .rightpanel::before {  
 position: absolute;  
 top: 0;  
 left: 0;  
 width: 10px;  
 height: 10px;  
 border-left: 2px solid #02a6b5;  
 border-top: 2px solid #02a6b5;  
 content: "";  
}  
.mainbox .rightpanel::after {  
 position: absolute;  
 top: 0;  
 right: 0;  
 width: 10px;  
 height: 10px;  
 border-right: 2px solid #02a6b5;  
 border-top: 2px solid #02a6b5;  
 content: "";  
}  
.mainbox .rightpanel .rightpanel-footer {  
 position: absolute;  
 left: 0;  
 bottom: 0;  
 width: 100%;  
}  
.mainbox .rightpanel .rightpanel-footer::before {  
 position: absolute;  
 bottom: 0;  
 left: 0;  
 content: "";  
 width: 10px;  
 height: 10px;  
 border-left: 2px solid #02a6b5;  
 border-bottom: 2px solid #02a6b5;  
}  
.mainbox .rightpanel .rightpanel-footer::after {  
 position: absolute;  
 bottom: 0;  
 right: 0;  
 content: "";  
 width: 10px;  
 height: 10px;  
 border-bottom: 2px solid #02a6b5;  
 border-right: 2px solid #02a6b5;  
}  
.mainbox .rightpanel h2 {  
 height: 0.6rem;  
 color: #fff;  
 line-height: 0.6rem;  
 text-align: center;  
 font-size: 0.25rem;  
 font-weight: 400;  
}  
.mainbox .rightpanel .chart {  
 height: 3rem;  
}  
/\* 约束屏幕尺寸 \*/  
@media screen and (max-width: 1024px) {  
 html {  
 font-size: 42px **!important**;  
 }  
}  
@media screen and (min-width: 1920px) {  
 html {  
 font-size: 80px **!important**;  
 }  
}

# surveillance.css

\* {  
 margin: 0;  
 padding: 0;  
 box-sizing: border-box;  
}  
li {  
 list-style: none;  
}  
/\* 声明字体\*/  
@font-face {  
 font-family: electronicFont;  
 src: url(../font/DS-DIGIT.TTF);  
}  
body {  
 background: url(../images/bg.jpg) no-repeat top center;  
 line-height: 1.15;  
}  
header {  
 position: relative;  
 height: 1.25rem;  
 background: url(../images/head\_bg.png) no-repeat;  
 background-size: 100% 100%;  
}  
header h1 {  
 font-size: 0.475rem;  
 color: #fff;  
 text-align: center;  
 line-height: 1rem;  
}  
header .showTime {  
 position: absolute;  
 top: 0;  
 right: 0.375rem;  
 line-height: 0.9375rem;  
 color: rgba(255, 255, 255, 0.7);  
 font-size: 0.25rem;  
}  
  
/\* Larger buttons styles \*/  
  
.button.large{  
 padding: 6px 15px;  
 text-transform: uppercase;  
}  
  
.button.large:active{  
 top: 2px;  
}  
  
/\* Colored buttons styles \*/  
  
.button.blue {  
 color: #fff;  
 text-shadow: 0 1px 0 rgba(0,0,0,.2);  
  
 background-image: -webkit-gradient(linear, left top, left bottom, from(rgba(255,255,255,.3)), to(rgba(255,255,255,0)));  
 background-image: -webkit-linear-gradient(top, rgba(255,255,255,.3), rgba(255,255,255,0));  
 background-image: -moz-linear-gradient(top, rgba(255,255,255,.3), rgba(255,255,255,0));  
 background-image: -ms-linear-gradient(top, rgba(255,255,255,.3), rgba(255,255,255,0));  
 background-image: -o-linear-gradient(top, rgba(255,255,255,.3), rgba(255,255,255,0));  
 background-image: linear-gradient(top, rgba(255,255,255,.3), rgba(255,255,255,0));  
}  
  
.button.blue{  
 background-color: #269CE9;  
 border-color: #269CE9;  
}  
  
.button.blue:hover{  
 background-color: #70B9E8;  
}  
  
.button.blue:active{  
 background: #269CE9;  
}  
  
.blue[disabled], .blue[disabled]:hover, .blue[disabled]:active{  
 border-color: #269CE9;  
 background: #269CE9;  
 color: #93D5FF;  
}  
  
.button.blueShadow {  
 color: #fff;  
 text-shadow: 0 1px 0 rgba(0,0,0,.2);  
  
 background-image: -webkit-gradient(linear, left top, left bottom, from(rgba(255,255,255,.3)), to(rgba(255,255,255,0)));  
 background-image: -webkit-linear-gradient(top, rgba(255,255,255,.3), rgba(255,255,255,0));  
 background-image: -moz-linear-gradient(top, rgba(255,255,255,.3), rgba(255,255,255,0));  
 background-image: -ms-linear-gradient(top, rgba(255,255,255,.3), rgba(255,255,255,0));  
 background-image: -o-linear-gradient(top, rgba(255,255,255,.3), rgba(255,255,255,0));  
 background-image: linear-gradient(top, rgba(255,255,255,.3), rgba(255,255,255,0));  
}  
  
.button.blueShadow{  
 background-color: White;  
 border-color: White;  
}  
  
.button.blueShadow:hover{  
 background-color: #70B9E8;  
}  
  
.button.blueShadow:active{  
 background: #93D5FF;  
}  
  
.blueShadow[disabled], .blueShadow[disabled]:hover, .blueShadow[disabled]:active{  
 border-color: #93D5FF;  
 background: #93D5FF;  
 color: #93D5FF;  
}  
  
/\* Group buttons \*/  
  
.button-group,  
.button-group li{  
 display: inline-block;  
 \*display: inline;  
 zoom: 1;  
 margin-right: -1px; /\* Overlap each right button border \*/  
}  
  
.button-group{  
 font-size: 0; /\* Inline block elements gap - fix \*/  
 margin: 0;  
 background: rgba(0, 0, 0, .1);  
 border-bottom: 3px rgba(0, 0, 0, .1);  
 padding: 3px;  
 -moz-border-radius: 3px;  
 -webkit-border-radius: 3px;  
 border-radius: 3px;  
}  
  
.button-group .button{  
 font-size: 13px; /\* Set the font size, different from inherited 0 \*/  
 -moz-border-radius: 0;  
 -webkit-border-radius: 0;  
 border-radius: 0;  
}  
  
.button-group .button:active{  
 -moz-box-shadow: 0 0 1px rgba(0, 0, 0, .2) inset, 5px 0 5px -3px rgba(0, 0, 0, .2) inset, -5px 0 5px -3px rgba(0, 0, 0, .2) inset;  
 -webkit-box-shadow: 0 0 1px rgba(0, 0, 0, .2) inset, 5px 0 5px -3px rgba(0, 0, 0, .2) inset, -5px 0 5px -3px rgba(0, 0, 0, .2) inset;  
 box-shadow: 0 0 1px rgba(0, 0, 0, .2) inset, 5px 0 5px -3px rgba(0, 0, 0, .2) inset, -5px 0 5px -3px rgba(0, 0, 0, .2) inset;  
}  
  
.button-group li:first-child .button{  
 -moz-border-radius: 3px 0 0 3px;  
 -webkit-border-radius: 3px 0 0 3px;  
 border-radius: 3px 0 0 3px;  
}  
  
.button-group li:first-child .button:active{  
 -moz-box-shadow: 0 0 1px rgba(0, 0, 0, .2) inset, -5px 0 5px -3px rgba(0, 0, 0, .2) inset;  
 -webkit-box-shadow: 0 0 1px rgba(0, 0, 0, .2) inset, -5px 0 5px -3px rgba(0, 0, 0, .2) inset;  
 box-shadow: 0 0 1px rgba(0, 0, 0, .2) inset, -5px 0 5px -3px rgba(0, 0, 0, .2) inset;  
}  
  
.button-group li:last-child .button{  
 -moz-border-radius: 0 3px 3px 0;  
 -webkit-border-radius: 0 3px 3px 0;  
 border-radius: 0 3px 3px 0;  
}  
  
.button-group li:last-child .button:active{  
 -moz-box-shadow: 0 0 1px rgba(0, 0, 0, .2) inset, 5px 0 5px -3px rgba(0, 0, 0, .2) inset;  
 -webkit-box-shadow: 0 0 1px rgba(0, 0, 0, .2) inset, 5px 0 5px -3px rgba(0, 0, 0, .2) inset;  
 box-shadow: 0 0 1px rgba(0, 0, 0, .2) inset, 5px 0 5px -3px rgba(0, 0, 0, .2) inset;  
}  
  
  
  
.mainbox {  
 display: flex;  
 min-width: 1024px;  
 max-width: 1920px;  
 margin: 0 auto;  
 padding: 0.125rem 0.125rem 0;  
}  
.mainbox .column {  
 flex: 3;  
}  
.mainbox .column:nth-child(2) {  
 flex: 5;  
 margin: 0 0.125rem 0.1875rem;  
 overflow: hidden;  
}  
.mainbox .panel {  
 position: relative;  
 height: 3.875rem;  
 padding: 0 0.1875rem 0.5rem;  
 margin-bottom: 0.1875rem;  
 border: 1px solid rgba(25, 186, 139, 0.17);  
 background: url(../images/line.png) rgba(255, 255, 255, 0.04);  
}  
.mainbox .panel::before {  
 position: absolute;  
 top: 0;  
 left: 0;  
 width: 10px;  
 height: 10px;  
 border-left: 2px solid #02a6b5;  
 border-top: 2px solid #02a6b5;  
 content: "";  
}  
.mainbox .panel::after {  
 position: absolute;  
 top: 0;  
 right: 0;  
 width: 10px;  
 height: 10px;  
 border-right: 2px solid #02a6b5;  
 border-top: 2px solid #02a6b5;  
 content: "";  
}  
.mainbox .panel .panel-footer {  
 position: absolute;  
 left: 0;  
 bottom: 0;  
 width: 100%;  
}  
.mainbox .panel .panel-footer::before {  
 position: absolute;  
 bottom: 0;  
 left: 0;  
 content: "";  
 width: 10px;  
 height: 10px;  
 border-left: 2px solid #02a6b5;  
 border-bottom: 2px solid #02a6b5;  
}  
.mainbox .panel .panel-footer::after {  
 position: absolute;  
 bottom: 0;  
 right: 0;  
 content: "";  
 width: 10px;  
 height: 10px;  
 border-bottom: 2px solid #02a6b5;  
 border-right: 2px solid #02a6b5;  
}  
.mainbox .panel h2 {  
 height: 0.6rem;  
 color: #fff;  
 line-height: 0.6rem;  
 text-align: center;  
 font-size: 0.25rem;  
 font-weight: 400;  
}  
.mainbox .panel .chart {  
 height: 3rem;  
}  
.no {  
 background-color: rgba(101, 132, 226, 0.1);  
 padding: 0.1875rem;  
}  
.no .no-hd {  
 position: relative;  
 border: 1px solid rgba(25, 186, 139, 0.17);  
}  
.no .no-hd::before {  
 position: absolute;  
 top: 0;  
 left: 0;  
 content: "";  
 width: 30px;  
 height: 10px;  
 border-top: 2px solid #02a6b5;  
 border-left: 2px solid #02a6b5;  
}  
.no .no-hd::after {  
 position: absolute;  
 bottom: 0;  
 right: 0;  
 content: "";  
 width: 30px;  
 height: 10px;  
 border-bottom: 2px solid #02a6b5;  
 border-right: 2px solid #02a6b5;  
}  
.no .no-hd ul {  
 display: flex;  
}  
.no .no-hd ul li {  
 position: relative;  
 flex: 1;  
 line-height: 1rem;  
 font-size: 0.875rem;  
 color: #ffeb7b;  
 text-align: center;  
 font-family: 'electronicFont';  
}  
.no .no-hd ul li::after {  
 content: "";  
 position: absolute;  
 top: 25%;  
 right: 0;  
 height: 50%;  
 width: 1px;  
 background-color: rgba(255, 255, 255, 0.2);  
}  
.no .no-bd ul {  
 display: flex;  
}  
.no .no-bd ul li {  
 flex: 1;  
 text-align: center;  
 color: rgba(255, 255, 255, 0.7);  
 font-size: 0.225rem;  
 height: 0.5rem;  
 line-height: 0.5rem;  
 padding-top: 10px;  
}  
.map {  
 position: relative;  
 height: 10.125rem;  
}  
.map .map1 {  
 width: 6.475rem;  
 height: 6.475rem;  
 position: absolute;  
 top: 50%;  
 left: 50%;  
 transform: translate(-50%, -50%);  
 background-size: 100% 100%;  
 opacity: 0.3;  
}  
.map .map2 {  
 width: 8.0375rem;  
 height: 8.0375rem;  
 position: absolute;  
 top: 50%;  
 left: 50%;  
 transform: translate(-50%, -50%);  
 background-size: 100% 100%;  
 animation: rotate1 15s linear infinite;  
 opacity: 0.6;  
}  
.map .map3 {  
 width: 7.075rem;  
 height: 7.075rem;  
 position: absolute;  
 top: 50%;  
 left: 50%;  
 transform: translate(-50%, -50%);  
 background-size: 100% 100%;  
 animation: rotate2 10s linear infinite;  
 opacity: 0.6;  
}  
.map .chart {  
 position: absolute;  
 top: 0;  
 left: 0;  
 width: 100%;  
 height: 10.125rem;  
 background-size: 100% 100%;  
}  
@keyframes rotate1 {  
 from {  
 transform: translate(-50%, -50%) rotate(0deg);  
 }  
 to {  
 transform: translate(-50%, -50%) rotate(360deg);  
 }  
}  
@keyframes rotate2 {  
 from {  
 transform: translate(-50%, -50%) rotate(0deg);  
 }  
 to {  
 transform: translate(-50%, -50%) rotate(-360deg);  
 }  
}  
/\* 约束屏幕尺寸 \*/  
@media screen and (max-width: 1024px) {  
 html {  
 font-size: 42px **!important**;  
 }  
}  
@media screen and (min-width: 1920px) {  
 html {  
 font-size: 80px **!important**;  
 }  
}

# MidBackend.js

import express from 'express';  
import bodyParser from 'body-parser';  
import \* as authDisplayGroup from './backendProcessor/authDisplayGroup.js';  
import \* as listenDisplayGroup from './backendProcessor/listenDisplayGroup.js';  
import mysql from 'mysql';  
  
import {mysqlConf} from '../../utils/info.js';  
export const c = mysql.createConnection(mysqlConf);  
await c.connect();  
const reconnectInterval = 60\*60000;//1h  
const pulseInterval = 60000;//1min  
  
setInterval(async function() {  
 c.end();  
 await c.connect();  
}, reconnectInterval);  
setInterval(() => c.ping(err => console.log('MySQL ping err:', err)), pulseInterval);  
  
  
/\*----------信息查询请求路由配置----------\*/  
async function UseMysql(req, res, handle) {  
  
 try{  
 await c.connect();  
 }  
 catch(e){  
 ***console***.log("Connect Release?");  
 res.send('数据库错误，请联系黄文伟');  
 res.end();  
 ***c***.end();  
 return;  
 }  
  
 let resJson = await handle(req, res);  
 res.send({'data': resJson});  
 res.end();  
 ***c***.end();  
}  
  
/\*----------信息查询请求路由配置----------\*/  
async function NoUseMysql(req, res, handle) {  
 let resJson = await handle(req, res);  
 res.send({'data': resJson});  
 res.end();  
}  
const authRouter = express.Router({  
 caseSensitive: false,// 不区分大小写  
});  
  
const listenRouter = express.Router({  
 caseSensitive: false,// 不区分大小写  
});  
// authRouter.get('/authRightRate', async function(req, res) {  
// await NoUseMysql(req, res, authDisplayGroup.handleAuthRightRate);  
// });  
// // localhost:9002/backend/authRightRate  
// authRouter.get('/authByCompany', async function(req, res) {  
// await NoUseMysql(req, res, authDisplayGroup.handleAuthByCompany);  
// });  
// // localhost:9002/backend/authByCompany  
  
authRouter.get('/certificateAmountEXchange', async function(req, res) {  
 await NoUseMysql(req, res, authDisplayGroup.handleCertificateAmountEXchange);  
});  
// localhost:9002/backend/certificateAmountEXchange  
  
authRouter.get('/certificateAmountGroupByWorkType', async function(req, res) {  
 await NoUseMysql(req, res, authDisplayGroup.handleCertificateAmountGroupByWorkType);  
});  
// localhost:9002/backend/certificateAmountGroupByWorkType  
  
authRouter.get('/certificateAmountGroupByWorkTypeEXchange', async function(req, res) {  
 await NoUseMysql(req, res, authDisplayGroup.handleCertificateAmountGroupByWorkTypeEXchange);  
});  
// localhost:9002/backend/certificateAmountGroupByWorkTypeEXchange  
  
authRouter.get('/copyRightAmountEXchange', async function(req, res) {  
 await NoUseMysql(req, res, authDisplayGroup.handleCopyRightAmountEXchange);  
});  
// localhost:9002/backend/copyRightAmountEXchange  
  
authRouter.get('/copyRightAmountGroupByIDtype', async function(req, res) {  
 await NoUseMysql(req, res, authDisplayGroup.handleCopyRightAmountGroupByIDtype);  
});  
// localhost:9002/backend/copyRightAmountGroupByIDtype  
  
authRouter.get('/copyRightAmountGroupByCopyrightType', async function(req, res) {  
 await NoUseMysql(req, res, authDisplayGroup.handleCopyRightAmountGroupByCopyrightType);  
});  
// localhost:9002/backend/copyRightAmountGroupByCopyrightType  
  
  
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
*/\*\*\*\* 监测维权 \*\*\*\*/*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
listenRouter.get('/TortCount', async function(req, res) {  
 await NoUseMysql(req, res, listenDisplayGroup.handleTortCount);  
});  
// localhost:9002/backend/listen/TortCount  
  
listenRouter.get('/TortClickCount', async function(req, res) {  
 await NoUseMysql(req, res, listenDisplayGroup.handleTortClickCount);  
});  
// localhost:9002/backend/listen/TortClickCount  
  
listenRouter.get('/TortCountEXchange', async function(req, res) {  
 await NoUseMysql(req, res, listenDisplayGroup.handleTortCountEXchange);  
});  
// localhost:9002/backend/listen/TortCountEXchange  
  
listenRouter.get('/TortCountGroupByWorkType', async function(req, res) {  
 await NoUseMysql(req, res, listenDisplayGroup.handleTortCountGroupByWorkType);  
});  
// localhost:9002/backend/listen/TortCountGroupByWorkType  
  
listenRouter.get('/TortCountGroupByCreationType', async function(req, res) {  
 await NoUseMysql(req, res, listenDisplayGroup.handleTortCountGroupByCreationType);  
});  
// localhost:9002/backend/listen/TortCountGroupByCreationType  
  
listenRouter.get('/TortCountGroupByWorkTypeEXchange', async function(req, res) {  
 await NoUseMysql(req, res, listenDisplayGroup.handleTortCountGroupByWorkTypeEXchange);  
});  
// localhost:9002/backend/listen/TortCountGroupByWorkTypeEXchange  
  
listenRouter.get('/TortCountGroupByCreationTypeEXchange', async function(req, res) {  
 await NoUseMysql(req, res, listenDisplayGroup.handleTortCountGroupByCreationTypeEXchange);  
});  
// localhost:9002/backend/listen/TortCountGroupByCreationTypeEXchange  
  
listenRouter.get('/TortCountGroupByTortSite', async function(req, res) {  
 await NoUseMysql(req, res, listenDisplayGroup.handleTortCountGroupByTortSite);  
});  
// localhost:9002/backend/listen/TortCountGroupByTortSite  
  
listenRouter.get('/TortCountGroupByTortSiteEXchange', async function(req, res) {  
 await NoUseMysql(req, res, listenDisplayGroup.handleTortCountGroupByTortSiteEXchange);  
});  
// localhost:9002/backend/listen/TortCountGroupByTortSiteEXchange  
  
listenRouter.get('/TortCountGroupByTortSiteGroupByWorkType', async function(req, res) {  
 await NoUseMysql(req, res, listenDisplayGroup.handleTortCountGroupByTortSiteGroupByWorkType);  
});  
// localhost:9002/backend/listen/TortCountGroupByTortSiteGroupByWorkType  
  
listenRouter.get('/Tort\_AND\_ClaimCountGroupByWorkType', async function(req, res) {  
 await NoUseMysql(req, res, listenDisplayGroup.handleTort\_AND\_ClaimCountGroupByWorkType);  
});  
// localhost:9002/backend/listen/Tort\_AND\_ClaimCountGroupByWorkType  
  
  
/\*----------http服务器配置----------\*/  
  
const app = express();  
const port = 9002;  
  
app.use(bodyParser.urlencoded({extended:false}));  
app.use(bodyParser.json());  
app.get('/backend1', function (req, res) {  
 res.send('Hello World');  
 })  
//app.js  
app.all('\*', function(req, res, next) {  
 res.header("Access-Control-Allow-Origin", "\*");  
 res.header("Access-Control-Allow-Headers", "X-Requested-With");  
 res.header("Access-Control-Allow-Methods","PUT,POST,GET,DELETE,OPTIONS");  
 res.header("X-Powered-By",' 3.2.1')  
 res.header("Content-Type", "application/json;charset=utf-8");  
 next();  
});  
app.use('/backend', authRouter);  
app.use('/backend/listen', listenRouter);  
  
  
/\*----------启动http服务器----------\*/  
  
app.listen(port, () => console.log(`MainMid listening on port ${port}!`));

# authDisplayGroup.js

*/\*\*  
 \** ***@file:*** *authDisplayGroup.js  
 \** ***@Description:*** *存证确权后端处理函数  
 \** ***@date*** *2021/5/23  
\*/*import ***sqlText*** from 'node-transform-mysql';  
  
import \* as mysqlUtils from '../../../utils/mysqlUtils.js';  
import \* as DateUtil from './DateUtil.js';  
import \* as localUtils from '../../../utils/localUtils.js';  
  
import ***util*** from 'util';  
  
import mysql from 'mysql';  
import {***c***} from "../MidBackend.js";  
import {***debugMode***, ***WORKTYPE***, ***CREATIONTYPE***} from '../../../utils/info.js';  
import {countGroupBy, countNumJoinRight, countNumJoinRightAll} from "./SelectUtil.js";  
const CONNECT = true;// When false, Send Random Response  
// export{  
// handleCertificateAmountEXchange,// 1） 存证总数量随时间的变化。  
// handleCertificateAmountGroupByWorkType,// 2） 截止当前不同作品类型。workType  
// handleCertificateAmountGroupByWorkTypeEXchange,// 3） 不同作品类型的存证数量随时间的变化。workType  
// handleCopyRightAmountEXchange,// 1） 版权通证总数量随时间的变化。 Amount  
// handleCopyRightAmountGroupByIDtype,// 1） 截止当前，在已生成的全部版权通证中，个人账户作为存证时的版权接收者 id\_type  
// handleCopyRightAmountGroupByCopyrightType// 1) 截止当前，不同类别通证的数量分布。 copyrightType Amount  
// }  
//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//  
// 新方案  
// 1.1 存证总数量随时间变化 折线图  
// 1.2 当前不同作品类型存证数量分布饼图  
// 1.3 最大的三种作品类型的存证数量随时间的变化  
// 2.1 版权通证总数量随时间变化 折线图  
// 3.1 个人账户与非个人账户接收者通证数量对比  
// 4.1 截止当前，不同类别通证的数量分布饼图  
//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//  
// 1. 存证信息-作品信息  
// 一维图（一个自变量）  
// 1） 存证总数量随时间的变化。  
export async function handleCertificateAmountEXchange(req, res) {  
  
 ***console***.time('handleCertificateAmountEXchange');  
 let sqlRes = await getCertificateAmountEXchange();  
 ***console***.timeEnd('handleCertificateAmountEXchange');  
 ***console***.log('--------------------');  
 return sqlRes;  
}  
async function getCertificateAmountEXchange() {  
 let [TimeStampArray,MonthArray] = DateUtil.getMonthTimeStampArray();  
 // console.log([TimeStampArray, MonthArray]);  
 let CertificateAmountEXchange = [];  
 for (let index = 0; index < 12; index++) {  
 let endTimeStamp = TimeStampArray[index];  
 let startTimeStamp = TimeStampArray[(index + 1)];  
 let valueRes = await countNumJoinRight("right\_token\_info", "work\_id", "work\_info", "work\_id",  
 endTimeStamp,startTimeStamp);  
 if(CONNECT == false)valueRes = localUtils.randomNumber(30,50);  
 let MonthInfo = {  
 "CertificateAmount": valueRes["num"],  
 "Month" : MonthArray[index + 1],  
 };  
 CertificateAmountEXchange.push(MonthInfo);  
 }  
 CertificateAmountEXchange.reverse();  
 ***console***.log(CertificateAmountEXchange);  
 return CertificateAmountEXchange;  
}  
  
// 2） 截止当前不同作品类型  
export async function handleCertificateAmountGroupByWorkType(req, res) {  
  
 ***console***.time('handleCertificateAmountGroupByWorkType');  
 let sqlRes = await getCertificateAmountGroupByWorkType();  
 ***console***.timeEnd('handleCertificateAmountGroupByWorkType');  
 ***console***.log('--------------------');  
 return sqlRes;  
}  
async function getCertificateAmountGroupByWorkType() {  
 let CertificateAmountGroupByWorkType = [];  
 let WorkTypeInfo = {};  
  
 if(CONNECT == true){  
 let Res = await countGroupBy("work\_info", "work\_type");  
 ***console***.log("Res:",Res);  
 let keys = ***Object***.keys(Res);  
 ***console***.log("keys:",keys);  
 for (let i = 0, n = keys.length, key; i < n; ++i) {  
 key = keys[i];  
 WorkTypeInfo = {  
 "workType":***WORKTYPE***[key],  
 "CertificateAmount":Res[key]  
 };  
 CertificateAmountGroupByWorkType.push(WorkTypeInfo);  
 }  
 ***console***.log("CertificateAmountGroupByWorkType:",CertificateAmountGroupByWorkType);  
 }  
 else{  
 CertificateAmountGroupByWorkType = [{  
 "workType":"音乐",  
 "CertificateAmount":0  
 },{  
 "workType":"电影",  
 "CertificateAmount":0  
 },{  
 "workType":"美术",  
 "CertificateAmount":0  
 }]  
 }  
  
 return CertificateAmountGroupByWorkType;  
}  
  
// 3） 不同作品类型的存证数量随时间的变化。workType  
export async function handleCertificateAmountGroupByWorkTypeEXchange(req, res) {  
  
 ***console***.time('handleCertificateAmountGroupByWorkTypeEXchange');  
 let sqlRes = await getCertificateAmountGroupByWorkTypeEXchange();  
  
  
 // let resJson = JSON.stringify(sqlRes);  
 ***console***.timeEnd('handleCertificateAmountGroupByWorkTypeEXchange');  
 ***console***.log('--------------------');  
 return sqlRes;  
}  
async function getCertificateAmountGroupByWorkTypeEXchange() {  
 let [TimeStampArray,MonthArray] = DateUtil.getSeasonTimeStampArray();  
 let CertificateAmountGroupByWorkTypeEXchange = [];  
 let CertificateAmountGroupByWorkType = [];  
 let SeasonGap = 1;  
 let WorkTypeInfo = {};  
 // console.log([TimeStampArray, MonthArray]);  
 if(CONNECT == true){  
 // *TODO 确定选中的类型* let index = 0;  
 let endTimeStamp = TimeStampArray[index];  
 let startTimeStamp = TimeStampArray[(index + 1)];  
 let keys = [];  
 while(keys.length<3 && index<3){  
 endTimeStamp = TimeStampArray[index];  
 startTimeStamp = TimeStampArray[(index + 1)];  
 let Res = await countGroupBy("work\_info", "work\_type", endTimeStamp ,startTimeStamp);  
 keys = ***Object***.keys(Res);  
 index = index + SeasonGap;  
 }  
  
 for (let index = 0; index < 4; index = index + SeasonGap) {  
 endTimeStamp = TimeStampArray[index];  
 startTimeStamp = TimeStampArray[(index + 1)];  
 let Res = await countGroupBy("work\_info", "work\_type",endTimeStamp ,startTimeStamp);  
 CertificateAmountGroupByWorkType =[];  
 for (let i = 0, n = keys.length, key; i < n; ++i) {  
 key = keys[i];  
 if(Res[key]==null)Res[key]=0;  
 let MonthInfo = {  
 "workType":***WORKTYPE***[key],  
 "CertificateAmount":Res[key],  
 "Month" : MonthArray[index],  
 };  
 CertificateAmountGroupByWorkType.push(MonthInfo);  
 }  
 CertificateAmountGroupByWorkTypeEXchange.push(CertificateAmountGroupByWorkType);  
  
 }  
 }  
 else{  
 for (let index = 0; index < 3; index = index + SeasonGap) {  
 let CertificateAmountGroupByWorkType = [];  
  
 WorkTypeInfo = {  
 "workType":"音乐",  
 "CertificateAmount":localUtils.randomNumber(80,100),  
 "Month" : MonthArray[index + SeasonGap],  
 };  
 CertificateAmountGroupByWorkType.push(WorkTypeInfo);  
 WorkTypeInfo = {  
 "workType":"电影",  
 "CertificateAmount":localUtils.randomNumber(60,80),  
 "Month" : MonthArray[index + SeasonGap],  
 };  
 CertificateAmountGroupByWorkType.push(WorkTypeInfo);  
 WorkTypeInfo = {  
 "workType":"美术",  
 "CertificateAmount":localUtils.randomNumber(40,60),  
 "Month" : MonthArray[index + SeasonGap],  
 };  
 CertificateAmountGroupByWorkType.push(WorkTypeInfo);  
 CertificateAmountGroupByWorkTypeEXchange.push(CertificateAmountGroupByWorkType);  
 }  
  
 }  
  
 CertificateAmountGroupByWorkTypeEXchange.reverse();  
 return CertificateAmountGroupByWorkTypeEXchange;  
}  
  
// 1） 版权通证总数量随时间的变化。 Amount  
export async function handleCopyRightAmountEXchange(req, res) {  
  
 ***console***.time('handleCopyRightAmountEXchange');  
 let sqlRes = await getCopyRightAmountEXchange();  
 ***console***.timeEnd('handleCopyRightAmountEXchange');  
 ***console***.log('--------------------');  
 return sqlRes;  
}  
async function getCopyRightAmountEXchange() {  
 let [TimeStampArray,MonthArray] = DateUtil.getMonthTimeStampArray();  
 let CopyRightAmountEXchange = [];  
 for (let index = 0; index < 12; index++) {  
 let endTimeStamp = TimeStampArray[index];  
 let startTimeStamp = TimeStampArray[(index + 1)];  
 let valueRes = await countNumJoinRightAll("right\_token\_info", "work\_id", "work\_info", "work\_id",  
 endTimeStamp,startTimeStamp);  
 let MonthInfo = {  
 "CopyRightAmount": valueRes["num"],  
 "Month" : MonthArray[index + 1],  
 };  
 CopyRightAmountEXchange.push(MonthInfo);  
 }  
 CopyRightAmountEXchange.reverse();  
 ***console***.log(CopyRightAmountEXchange);  
 return CopyRightAmountEXchange;  
}  
  
// 1） 截止当前，在已生成的全部版权通证中，个人账户作为存证时的版权接收者（版权持有者证件类型为居民身份证、军官证与护照）  
// 与非个人账户作为存证时的版权接收者（版权持有者证件类型为营业执照、企业法人营业执照、组织机构代码证书、事业单位法人证书、社团法人证书、其他有效证件）  
// 的通证数量分布。 id\_type  
// 1..9 1.2.4为个人  
export async function handleCopyRightAmountGroupByIDtype(req, res) {  
  
 ***console***.time('handleCopyRightAmountGroupByIDtype');  
 let sqlRes = await getCopyRightAmountGroupByIDtype();  
 ***console***.timeEnd('handleCopyRightAmountGroupByIDtype');  
 ***console***.log('--------------------');  
 return sqlRes;  
}  
async function getCopyRightAmountGroupByIDtype() {  
 let CopyRightAmountGroupByIDtype = {};  
 if(CONNECT == false){  
 CopyRightAmountGroupByIDtype = {  
 "个人账户数目" : localUtils.randomNumber(300,500),  
 "非个人账户数目": localUtils.randomNumber(600,1000),  
 };  
 }  
 else{  
 let sqlRight =util.format(  
 'SELECT\n' +  
 '\tCOUNT(right\_token\_info.copyright\_id) AS num\n' +  
 'FROM\n' +  
 '\tright\_token\_info\n' +  
 'WHERE\n' +  
 '\tright\_token\_info.id\_type = 3 OR \n' +  
 '\tright\_token\_info.id\_type = 5 OR \n' +  
 '\tright\_token\_info.id\_type = 6 OR \n' +  
 '\tright\_token\_info.id\_type = 7 OR \n' +  
 '\tright\_token\_info.id\_type = 8 OR \n' +  
 '\tright\_token\_info.id\_type = 9');  
 ***console***.log(sqlRight);  
 let sqlRes = await mysqlUtils.sql(***c***, sqlRight);  
 ***console***.log(sqlRes);  
 let InPersonalNum = 0;  
 sqlRes.forEach(function(item,index){  
 InPersonalNum = item['num'];  
 });  
 sqlRight =util.format(  
 'SELECT\n' +  
 '\tCOUNT(right\_token\_info.copyright\_id) AS num\n' +  
 'FROM\n' +  
 '\tright\_token\_info\n' +  
 'WHERE\n' +  
 '\tright\_token\_info.id\_type = 1 OR \n' +  
 '\tright\_token\_info.id\_type = 2 OR \n' +  
 '\tright\_token\_info.id\_type = 4');  
 ***console***.log(sqlRight);  
 sqlRes = await mysqlUtils.sql(***c***, sqlRight);  
 ***console***.log(sqlRes);  
 let PersonalNum = 0;  
 sqlRes.forEach(function(item,index){  
 PersonalNum = item['num'];  
 });  
  
 CopyRightAmountGroupByIDtype = {  
 "个人账户数目" : PersonalNum,  
 "非个人账户数目": InPersonalNum,  
 };  
 }  
 return CopyRightAmountGroupByIDtype;  
}  
  
// 二维图（两个自变量）  
// 2） 不同类型的账户作为版权通证接收者的通证数量随时间的变化。 IDType NeedTime  
// 4. 版权信息-copyrightType  
// 一维图（一个自变量）  
// 1) 截止当前，不同类别通证的数量分布。 copyrightType Amount  
export async function handleCopyRightAmountGroupByCopyrightType(req, res) {  
  
 ***console***.time('handleCopyRightAmountGroupByCopyrightType');  
 let sqlRes = await getCopyRightAmountGroupByCopyrightType();  
 ***console***.timeEnd('handleCopyRightAmountGroupByCopyrightType');  
 ***console***.log('--------------------');  
 return sqlRes;  
}  
async function getCopyRightAmountGroupByCopyrightType() {  
 let CopyRightAmountGroupByIDtype = {};  
 if(CONNECT == true){  
 let sqlRight =util.format(  
 'SELECT\n' +  
 '\tCOUNT(right\_token\_info.copyright\_id) AS num, \n' +  
 '\tright\_token\_info.copyright\_type\n' +  
 'FROM\n' +  
 '\tright\_token\_info\n' +  
 'GROUP BY\n' +  
 '\tright\_token\_info.copyright\_type\n' +  
 'ORDER BY\n' +  
 '\tright\_token\_info.copyright\_type');  
 ***console***.log(sqlRight);  
 let sqlRes = await mysqlUtils.sql(***c***, sqlRight);  
 ***console***.log(sqlRes);  
 let AmountGroup = {};  
 sqlRes.forEach(function(item,index){  
 AmountGroup[index] = item['num'];  
 });  
 ***console***.log(AmountGroup);  
 CopyRightAmountGroupByIDtype = {  
 "复制权" : AmountGroup[0],  
 "发行权" : AmountGroup[1],  
 "出租权" : AmountGroup[2],  
 "展览权" : AmountGroup[3],  
 "表演权" : AmountGroup[4],  
 "放映权" : AmountGroup[5],  
 "广播" : AmountGroup[6],  
 "信息网络传播权" : AmountGroup[7],  
 "摄制权" : AmountGroup[8],  
 "改编权" : AmountGroup[9],  
 "翻译权" : AmountGroup[10],  
 "汇编权" : AmountGroup[11],  
 "其他" : AmountGroup[12]  
 }  
 }  
 else {  
 CopyRightAmountGroupByIDtype = {  
 "复制权" : localUtils.randomNumber(200,500),  
 "发行权" : localUtils.randomNumber(200,500),  
 "出租权" : localUtils.randomNumber(200,500),  
 "展览权" : localUtils.randomNumber(200,500),  
 "表演权" : localUtils.randomNumber(200,500),  
 "放映权" : localUtils.randomNumber(200,500),  
 "广播" : localUtils.randomNumber(200,500),  
 "信息网络传播权" : localUtils.randomNumber(200,500),  
 "摄制权" : localUtils.randomNumber(200,500),  
 "改编权" : localUtils.randomNumber(200,500),  
 "翻译权" : localUtils.randomNumber(200,500),  
 "汇编权" : localUtils.randomNumber(200,500),  
 "其他" : localUtils.randomNumber(50,200),  
  
 }  
 }  
  
 return CopyRightAmountGroupByIDtype;  
}

# listenDisplayGroup.js

*/\*\*  
 \** ***@file:*** *listenDisplayGroup.js  
 \** ***@Description:*** *监测维权后端处理函数  
 \** ***@date*** *2021/5/23  
\*/*import ***sqlText*** from 'node-transform-mysql';  
  
import \* as mysqlUtils from '../../../utils/mysqlUtils.js';  
import \* as DateUtil from './DateUtil.js';  
import \* as localUtils from '../../../utils/localUtils.js';  
import {countGroupBy, countNum, countNumJoinRight} from "./SelectUtil.js";  
  
import ***util*** from 'util';  
  
import mysql from 'mysql';  
import {***c***} from "../MidBackend.js";  
import {***debugMode***, ***WORKTYPE***, ***CREATIONTYPE***, ***TORTSITE***} from '../../../utils/info.js';  
  
const CONNECT = true;// When false, Send Random Response  
// export{  
// handleTortCount,// 发现的侵权总数量  
// handleTortClickCount,// 发现的侵权的总点击次数  
// handleTortCountEXchange,// 发现的侵权总数量随时间的变化  
// handleTortCountGroupByWorkType,// 截止当前不同作品类型或者不同作品类型下发生的侵权数量分布  
// handleTortCountGroupByCreationType,// 截止当前不同作品类型或者不同创作类型下发生的侵权数量分布  
// handleTortCountGroupByWorkTypeEXchange,// 不同作品类型的侵权数量随时间的变化  
// handleTortCountGroupByCreationTypeEXchange,// 不同创作类型的侵权数量随时间的变化  
// handleTortCountGroupByTortSite,// 截止当前，在前N个侵权站点，发现的侵权数量分布  
// handleTortCountGroupByTortSiteEXchange,// 截止当前，在前N个侵权站点，发现的侵权数量分布。  
// handleTortCountGroupByTortSiteGroupByWorkType,// 截止当前，前N个侵权站点发现的侵权数量在不同作品类型下的分布  
// handleTort\_AND\_ClaimCountGroupByWorkType,// 截止当前，在不同作品类型下的侵权维权分布  
// }  
/\*  
 \* @param req: 请求  
 \* @param res: 返回  
 \* @date: 2021/6/2 17:08  
 \* @description:发现的侵权总数量。  
 \*/  
export async function handleTortCount(req, res) {  
 ***console***.time('handleTortCount');  
 let sqlRes = await getTortCount();  
 ***console***.timeEnd('handleTortCount');  
 ***console***.log('--------------------');  
 return sqlRes;  
}  
async function getTortCount() {  
 let tortCount = 0;  
 if(CONNECT == true){  
 let Res = await countNum("tort\_info","sample\_id");  
 tortCount = Res['num'];  
 }  
 else{  
 tortCount = 0;  
 }  
 ***console***.log("tortCount =",tortCount);  
 return tortCount;  
}  
  
/\*  
 \* @param req: 请求  
 \* @param res: 返回  
 \* @date: 2021/6/2 17:08  
 \* @description:发现的侵权总数量的总点击次数。  
 \*/  
export async function handleTortClickCount(req, res) {  
 ***console***.time('handleTortClickCount');  
 let sqlRes = await getTortClickCount();  
 ***console***.timeEnd('handleTortClickCount');  
 ***console***.log('--------------------');  
 return sqlRes;  
}  
async function getTortClickCount() {  
 let TortClickCount = 0;  
 if(CONNECT == true){  
 let sqlRight = "SELECT\n" +  
 "\tSUM(Type.click\_count) AS num\n" +  
 "FROM\n" +  
 "\t(\n" +  
 "\t\tSELECT DISTINCT\n" +  
 "\t\t\ttort\_info.work\_id, \n" +  
 "\t\t\ttort\_info.click\_count\n" +  
 "\t\tFROM\n" +  
 "\t\t\ttort\_info\n" +  
 "\t) AS Type";  
 let sqlRes = await mysqlUtils.sql(***c***, sqlRight);  
 sqlRes.forEach(value =>  
 TortClickCount = value['num']  
 );  
 }  
 else{  
 TortClickCount = 0;  
 }  
 ***console***.log("TortClickCount =",TortClickCount);  
 return TortClickCount;  
}  
  
/\*  
 \* @param req: 请求  
 \* @param res: 返回  
 \* @date: 2021/5/25 17:08  
 \* @description:发现的侵权总数量随时间的变化。  
 \*/  
export async function handleTortCountEXchange(req, res) {  
  
 ***console***.time('handleTortCountEXchange');  
 let sqlRes = await getTortCountEXchange();  
 console.timeEnd('handleTortCountEXchange');  
 console.log('--------------------');  
 return sqlRes;  
}  
async function getTortCountEXchange() {  
 let [TimeStampArray,MonthArray] = DateUtil.getMonthTimeStampArray();  
 let TortCountEXchange = [];  
 let valueRes = 0;  
 for (let index = 0; index < 12; index++) {  
 let endTimeStamp = TimeStampArray[index];  
 let startTimeStamp = TimeStampArray[(index + 1)];  
 let sqlRight = util.format('SELECT DISTINCT\n' +  
 '\tCOUNT(tort\_info.work\_id) AS num\n' +  
 'FROM\n' +  
 '\ttort\_info\n' +  
 '\t\tWHERE\n' +  
 '\t\t\ttort\_info.monitor\_time <= %s AND\n' +  
 '\t\t\ttort\_info.monitor\_time > %s\n',endTimeStamp,startTimeStamp)  
 let sqlRes = await mysqlUtils.sql(c, sqlRight);  
 sqlRes.forEach(value =>  
 valueRes = value['num']  
 );  
 if(CONNECT == false)valueRes = localUtils.randomNumber(30,50);  
 let MonthInfo = {  
 "TortCount": valueRes,  
 "Month" : MonthArray[index + 1],  
 };  
 TortCountEXchange.push(MonthInfo);  
 }  
 TortCountEXchange.reverse();  
 console.log(TortCountEXchange);  
 return TortCountEXchange;  
}  
  
/\*  
 \* @param req: 请求  
 \* @param res: 返回  
 \* @return: WorkType  
 \* @date: 2021/5/25 17:31  
 \* @description:截止当前不同作品类型或者不同作品类型下发生的侵权数量分布。  
 \*/  
export async function handleTortCountGroupByWorkType(req, res) {  
  
 console.time('handleTortCountGroupByWorkType');  
 let sqlRes = await getTortCountGroupByWorkType();  
 console.timeEnd('handleTortCountGroupByWorkType');  
 console.log('--------------------');  
 return sqlRes;  
}  
async function getTortCountGroupByWorkType() {  
 let TortCountGroupByWorkType = [];  
 let WorkTypeInfo = {};  
 if(CONNECT == true){  
 let sqlRight = util.format("SELECT\n" +  
 "\tType.work\_type, \n" +  
 "\tCOUNT(Type.work\_id) AS num\n" +  
 "FROM\n" +  
 "\t(\n" +  
 "\t\tSELECT DISTINCT\n" +  
 "\t\t\ttort\_info.work\_id, \n" +  
 "\t\t\twork\_info.work\_type\n" +  
 "\t\tFROM\n" +  
 "\t\t\ttort\_info\n" +  
 "\t\t\tINNER JOIN\n" +  
 "\t\t\twork\_info\n" +  
 "\t\t\tON \n" +  
 "\t\t\t\ttort\_info.work\_id = work\_info.work\_id\n" +  
 "\t) AS Type\n" +  
 "GROUP BY\n" +  
 "\tType.work\_type\n" +  
 "ORDER BY\n" +  
 "\tnum DESC\n" +  
 "LIMIT 6");  
 let sqlRes = await mysqlUtils.sql(c, sqlRight);  
 sqlRes.forEach(function(item){  
 let WorkTypeInfo = {  
 "workType":WORKTYPE[item['work\_type']],  
 "TortCount":item['num'],  
 };  
 TortCountGroupByWorkType.push(WorkTypeInfo);  
 });  
  
 // 空数据库保护  
 if(TortCountGroupByWorkType.length < 6){  
 PROTOTECT\_getTortCountGroupByWorkType(TortCountGroupByWorkType);  
 }  
 }  
 else{  
 PROTOTECT\_getTortCountGroupByWorkType(TortCountGroupByWorkType);  
 }  
 console.log(TortCountGroupByWorkType);  
 return TortCountGroupByWorkType;  
}  
function PROTOTECT\_getTortCountGroupByWorkType(TortCountGroupByWorkType) {  
 console.log("PROTOTECT\_getTortCountGroupByWorkType");  
  
 let WorkTypeInfo = {};  
 WorkTypeInfo = {  
 "workType":WORKTYPE["1"],  
 "TortCount":0  
 };  
 TortCountGroupByWorkType.push(WorkTypeInfo);  
 WorkTypeInfo = {  
 "workType":WORKTYPE["2"],  
 "TortCount":0  
 };  
 TortCountGroupByWorkType.push(WorkTypeInfo);  
 WorkTypeInfo = {  
 "workType":WORKTYPE["3"],  
 "TortCount":0  
 };  
 TortCountGroupByWorkType.push(WorkTypeInfo);  
 WorkTypeInfo = {  
 "workType":WORKTYPE["4"],  
 "TortCount":0  
 };  
 TortCountGroupByWorkType.push(WorkTypeInfo);  
 WorkTypeInfo = {  
 "workType":WORKTYPE["5"],  
 "TortCount":0  
 };  
 TortCountGroupByWorkType.push(WorkTypeInfo);  
 WorkTypeInfo = {  
 "workType":WORKTYPE["6"],  
 "TortCount":0  
 };  
 TortCountGroupByWorkType.push(WorkTypeInfo);  
 return TortCountGroupByWorkType;  
}  
/\*  
 \* @param req: 请求  
 \* @param res: 返回  
 \* @return: null  
 \* @date: 2021/5/25 17:31  
 \* @description:截止当前不同作品类型或者不同创作类型下发生的侵权数量分布。  
 \*/  
export async function handleTortCountGroupByCreationType(req, res) {  
  
 console.time('handleTortCountGroupByCreationType');  
 let sqlRes = await getTortCountGroupByCreationType();  
 console.timeEnd('handleTortCountGroupByCreationType');  
 console.log('--------------------');  
 return sqlRes;  
}  
async function getTortCountGroupByCreationType() {  
 let TortCountGroupByCreationType = [];  
  
  
 if(CONNECT == true){  
 let sqlRight = util.format("SELECT\n" +  
 "\tType.creation\_type, \n" +  
 "\tCOUNT(Type.work\_id) AS num\n" +  
 "FROM\n" +  
 "\t(\n" +  
 "\t\tSELECT DISTINCT\n" +  
 "\t\t\ttort\_info.work\_id, \n" +  
 "\t\t\twork\_info.creation\_type\n" +  
 "\t\tFROM\n" +  
 "\t\t\ttort\_info\n" +  
 "\t\t\tINNER JOIN\n" +  
 "\t\t\twork\_info\n" +  
 "\t\t\tON \n" +  
 "\t\t\t\ttort\_info.work\_id = work\_info.work\_id\n" +  
 "\t) AS Type\n" +  
 "GROUP BY\n" +  
 "\tType.creation\_type\n" +  
 "ORDER BY\n" +  
 "\tnum DESC\n" +  
 "LIMIT 3");  
 let sqlRes = await mysqlUtils.sql(c, sqlRight);  
 sqlRes.forEach(function(item){  
 let CreationTypeInfo = {  
 "creationType":WORKTYPE[item['creation\_type']],  
 "TortCount":item['num'],  
 };  
 TortCountGroupByCreationType.push(CreationTypeInfo);  
 });  
  
 // 空数据库保护  
 if(TortCountGroupByCreationType.length < 3){  
 PROTECT\_getTortCountGroupByCreationType(TortCountGroupByCreationType);  
 }  
  
  
 }  
 else{  
 PROTECT\_getTortCountGroupByCreationType(TortCountGroupByCreationType);  
 }  
 console.log(TortCountGroupByCreationType);  
 return TortCountGroupByCreationType;  
}  
function PROTECT\_getTortCountGroupByCreationType(TortCountGroupByCreationType){  
 console.log("PROTECT\_getTortCountGroupByCreationType");  
  
 let CreationTypeInfo={};  
 CreationTypeInfo = {  
 "creationType":CREATIONTYPE["3"],  
 "TortCount":0  
 };  
 TortCountGroupByCreationType.push(CreationTypeInfo);  
 CreationTypeInfo = {  
 "creationType":CREATIONTYPE["2"],  
 "TortCount":0  
 };  
 TortCountGroupByCreationType.push(CreationTypeInfo);  
 CreationTypeInfo = {  
 "creationType":CREATIONTYPE["1"],  
 "TortCount":0  
 };  
 TortCountGroupByCreationType.push(CreationTypeInfo);  
}  
/\*  
 \* @param req: 请求  
 \* @param res: 返回  
 \* @return: WorkType  
 \* @date: 2021/5/25 17:31  
 \* @description:3） 不同作品类型的侵权数量随时间的变化。  
 \*/  
export async function handleTortCountGroupByWorkTypeEXchange(req, res) {  
  
 console.time('handleTortCountGroupByWorkTypeEXchange');  
 let sqlRes = await getTortCountGroupByWorkTypeEXchange();  
 console.timeEnd('handleTortCountGroupByWorkTypeEXchange');  
 console.log('--------------------');  
 return sqlRes;  
}  
async function getTortCountGroupByWorkTypeEXchange() {  
 let TortCountGroupByWorkTypeEXchange = [];  
 let [TimeStampArray,MonthArray] = DateUtil.getMonthTimeStampArray();  
 let MonthGap = 1;  
 let WorkTypeInfo = {};  
 let TortCountGroupByWorkType = [];  
 let endTimeStamp = TimeStampArray[0];  
 let startTimeStamp = TimeStampArray[1];  
 if(CONNECT == true){  
 // TODO 确定选中的类型  
 let keys = [];  
 let index = 0;  
 while(keys.length < 3 && index < 12){  
 keys=[]  
 endTimeStamp = TimeStampArray[index];  
 startTimeStamp = TimeStampArray[(index + 1)];  
 let sqlRight = util.format("SELECT\n" +  
 "\tType.work\_type, \n" +  
 "\tCOUNT(Type.work\_id) AS num\n" +  
 "FROM\n" +  
 "\t(\n" +  
 "\t\tSELECT DISTINCT\n" +  
 "\t\t\ttort\_info.work\_id, \n" +  
 "\t\t\twork\_info.work\_type\n" +  
 "\t\tFROM\n" +  
 "\t\t\ttort\_info\n" +  
 "\t\t\tINNER JOIN\n" +  
 "\t\t\twork\_info\n" +  
 "\t\t\tON \n" +  
 "\t\t\t\ttort\_info.work\_id = work\_info.work\_id\n" +  
 "\t\tWHERE\n" +  
 "\t\t\ttort\_info.monitor\_time <= %s AND\n" +  
 "\t\t\ttort\_info.monitor\_time > %s\n" +  
 "\t) AS Type\n" +  
 "GROUP BY\n" +  
 "\tType.work\_type\n" +  
 "ORDER BY\n" +  
 "\tnum DESC\n" +  
 "LIMIT 3",endTimeStamp,startTimeStamp);  
 let sqlRes = await mysqlUtils.sql(c, sqlRight);  
 sqlRes.forEach(function(item){  
 keys.push(item['work\_type']);  
 });  
 index = index + MonthGap  
 }  
 if(keys.length < 3){  
 \_PROTECT\_getTortCountGroupByWorkTypeEXchange(TortCountGroupByWorkTypeEXchange,MonthGap,MonthArray);  
 TortCountGroupByWorkTypeEXchange.reverse();  
 return TortCountGroupByWorkTypeEXchange;  
 }  
 for (let index = 0; index < 12; index = index + MonthGap) {  
 endTimeStamp = TimeStampArray[index];  
 startTimeStamp = TimeStampArray[(index + 1)];  
 let TortCountGroupByWorkType = [];  
 let sqlRight = util.format("SELECT\n" +  
 "\tType.work\_type, \n" +  
 "\tCOUNT(Type.work\_id) AS num\n" +  
 "FROM\n" +  
 "\t(\n" +  
 "\t\tSELECT DISTINCT\n" +  
 "\t\t\ttort\_info.work\_id, \n" +  
 "\t\t\twork\_info.work\_type\n" +  
 "\t\tFROM\n" +  
 "\t\t\ttort\_info\n" +  
 "\t\t\tINNER JOIN\n" +  
 "\t\t\twork\_info\n" +  
 "\t\t\tON \n" +  
 "\t\t\t\ttort\_info.work\_id = work\_info.work\_id\n" +  
 "\t\tWHERE\n" +  
 "\t\t\ttort\_info.monitor\_time <= %s AND\n" +  
 "\t\t\ttort\_info.monitor\_time > %s\n" +  
 "\t) AS Type\n" +  
 "GROUP BY\n" +  
 "\tType.work\_type\n" +  
 "ORDER BY\n" +  
 "\tnum DESC\n" +  
 "LIMIT 3",endTimeStamp,startTimeStamp);  
 let sqlRes = await mysqlUtils.sql(c, sqlRight);  
 let Res = {};  
 sqlRes.forEach(function(item){  
 Res[item['work\_type']]=item['num']  
 });  
 for (let i = 0, n = keys.length, key; i < n; ++i) {  
 key = keys[i];  
 if(Res[key]==null)Res[key]=0;  
 let MonthInfo = {  
 "workType":WORKTYPE[key],  
 "TortCount":Res[key],  
 "Month" : MonthArray[index + MonthGap],  
 };  
 TortCountGroupByWorkType.push(MonthInfo);  
 };  
 TortCountGroupByWorkTypeEXchange.push(TortCountGroupByWorkType);  
 }  
 }  
 else{  
 \_PROTECT\_getTortCountGroupByWorkTypeEXchange(TortCountGroupByWorkTypeEXchange,MonthGap,MonthArray);  
 }  
 TortCountGroupByWorkTypeEXchange.reverse();  
 console.log(TortCountGroupByWorkTypeEXchange);  
 return TortCountGroupByWorkTypeEXchange;  
}  
function \_PROTECT\_getTortCountGroupByWorkTypeEXchange(TortCountGroupByWorkTypeEXchange,MonthGap,MonthArray) {  
 console.log("\_PROTECT\_getTortCountGroupByWorkTypeEXchange");  
  
 for (let index = 0; index < 12; index = index + MonthGap) {  
 let TortCountGroupByWorkType = [];  
 let WorkTypeInfo = {};  
 WorkTypeInfo = {  
 "workType":WORKTYPE["1"],  
 "TortCount":0,  
 "Month" : MonthArray[index + MonthGap],  
 };  
 TortCountGroupByWorkType.push(WorkTypeInfo);  
 WorkTypeInfo = {  
 "workType":WORKTYPE["2"],  
 "TortCount":0,  
 "Month" : MonthArray[index + MonthGap],  
 };  
 TortCountGroupByWorkType.push(WorkTypeInfo);  
 WorkTypeInfo = {  
 "workType":WORKTYPE["3"],  
 "TortCount":0,  
 "Month" : MonthArray[index + MonthGap],  
 };  
 TortCountGroupByWorkType.push(WorkTypeInfo);  
 TortCountGroupByWorkTypeEXchange.push(TortCountGroupByWorkType);  
 }  
}  
/\*  
 \* @param req: 请求  
 \* @param res: 返回  
 \* @return: null  
 \* @date: 2021/5/25 17:31  
 \* @description:不同创作类型的侵权数量随时间的变化。  
 \*/  
export async function handleTortCountGroupByCreationTypeEXchange(req, res) {  
  
 console.time('handleTortCountGroupByCreationTypeEXchange');  
 let sqlRes = await getTortCountGroupByCreationTypeEXchange();  
 console.timeEnd('handleTortCountGroupByCreationTypeEXchange');  
 console.log('--------------------');  
 return sqlRes;  
}  
async function getTortCountGroupByCreationTypeEXchange() {  
 let TortCountGroupByCreationTypeEXchange = [];  
 let [TimeStampArray,MonthArray] = DateUtil.getMonthTimeStampArray();  
 let MonthGap = 1;  
 let index = 0;  
 let endTimeStamp = TimeStampArray[index];  
 let startTimeStamp = TimeStampArray[(index + 1)];  
 if(CONNECT == true){  
 // TODO 确定选中的类型  
 let keys = [];  
 while(keys.length<3 && index<12){  
 keys = [];  
 endTimeStamp = TimeStampArray[index];  
 startTimeStamp = TimeStampArray[(index + 1)];  
 let sqlRight = util.format("SELECT\n" +  
 "\tType.creation\_type, \n" +  
 "\tCOUNT(Type.work\_id) AS num\n" +  
 "FROM\n" +  
 "\t(\n" +  
 "\t\tSELECT DISTINCT\n" +  
 "\t\t\ttort\_info.work\_id, \n" +  
 "\t\t\twork\_info.creation\_type\n" +  
 "\t\tFROM\n" +  
 "\t\t\ttort\_info\n" +  
 "\t\t\tINNER JOIN\n" +  
 "\t\t\twork\_info\n" +  
 "\t\t\tON \n" +  
 "\t\t\t\ttort\_info.work\_id = work\_info.work\_id\n" +  
 "\t\tWHERE\n" +  
 "\t\t\ttort\_info.monitor\_time <= %s AND\n" +  
 "\t\t\ttort\_info.monitor\_time > %s\n" +  
 "\t) AS Type\n" +  
 "GROUP BY\n" +  
 "\tType.creation\_type\n" +  
 "ORDER BY\n" +  
 "\tnum DESC\n" +  
 "LIMIT 3",endTimeStamp,startTimeStamp);  
 let sqlRes = await mysqlUtils.sql(c, sqlRight);  
 sqlRes.forEach(function(item){  
 keys.push(item['creation\_type']);  
 });  
 index = index + MonthGap;  
 }  
 if(keys.length < 3){  
 \_PROTECT\_getTortCountGroupByCreationTypeEXchange(TortCountGroupByCreationTypeEXchange, MonthGap, MonthArray)  
 TortCountGroupByCreationTypeEXchange.reverse();  
 return TortCountGroupByCreationTypeEXchange;  
 }  
  
  
  
 for (let index = 0; index < 12; index = index + MonthGap) {  
 endTimeStamp = TimeStampArray[index];  
 startTimeStamp = TimeStampArray[(index + 1)];  
 let TortCountGroupByCreationType = [];  
 let sqlRight = util.format("SELECT\n" +  
 "\tType.creation\_type, \n" +  
 "\tCOUNT(Type.work\_id) AS num\n" +  
 "FROM\n" +  
 "\t(\n" +  
 "\t\tSELECT DISTINCT\n" +  
 "\t\t\ttort\_info.work\_id, \n" +  
 "\t\t\twork\_info.creation\_type\n" +  
 "\t\tFROM\n" +  
 "\t\t\ttort\_info\n" +  
 "\t\t\tINNER JOIN\n" +  
 "\t\t\twork\_info\n" +  
 "\t\t\tON \n" +  
 "\t\t\t\ttort\_info.work\_id = work\_info.work\_id\n" +  
 "\t\tWHERE\n" +  
 "\t\t\ttort\_info.monitor\_time <= %s AND\n" +  
 "\t\t\ttort\_info.monitor\_time > %s\n" +  
 "\t) AS Type\n" +  
 "GROUP BY\n" +  
 "\tType.creation\_type\n" +  
 "ORDER BY\n" +  
 "\tnum DESC\n" +  
 "LIMIT 3",endTimeStamp,startTimeStamp);  
 let sqlRes = await mysqlUtils.sql(c, sqlRight);  
 let Res = {};  
 sqlRes.forEach(function(item){  
 Res[item['creation\_type']]=item['num']  
 });  
 for (let i = 0, n = keys.length, key; i < n; ++i) {  
 key = keys[i];  
 if(Res[key]==null)Res[key]=0;  
 let MonthInfo = {  
 "creationType":WORKTYPE[key],  
 "TortCount":Res[key],  
 "Month" : MonthArray[index + MonthGap],  
 };  
 TortCountGroupByCreationType.push(MonthInfo);  
 };  
 TortCountGroupByCreationTypeEXchange.push(TortCountGroupByCreationType);  
 }  
 }  
 else{  
 \_PROTECT\_getTortCountGroupByCreationTypeEXchange(TortCountGroupByCreationTypeEXchange, MonthGap, MonthArray)  
 }  
 TortCountGroupByCreationTypeEXchange.reverse();  
 console.log(TortCountGroupByCreationTypeEXchange);  
 return TortCountGroupByCreationTypeEXchange;  
}  
function \_PROTECT\_getTortCountGroupByCreationTypeEXchange(TortCountGroupByCreationTypeEXchange, MonthGap, MonthArray) {  
 console.log("\_PROTECT\_getTortCountGroupByCreationTypeEXchange");  
  
 for (let index = 0; index < 12; index = index + MonthGap) {  
 let TortCountGroupByWorkType = [];  
 let MonthInfo = {};  
 MonthInfo = {  
 "creationType":CREATIONTYPE["1"],  
 "TortCount":0,  
 "Month" : MonthArray[index + MonthGap],  
 };  
 TortCountGroupByWorkType.push(MonthInfo);  
 MonthInfo = {  
 "creationType":CREATIONTYPE["2"],  
 "TortCount":0,  
 "Month" : MonthArray[index + MonthGap],  
 };  
 TortCountGroupByWorkType.push(MonthInfo);  
 MonthInfo = {  
 "creationType":CREATIONTYPE["3"],  
 "TortCount":0,  
 "Month" : MonthArray[index + MonthGap],  
 };  
 TortCountGroupByWorkType.push(MonthInfo);  
 TortCountGroupByCreationTypeEXchange.push(TortCountGroupByWorkType);  
 }  
}  
/\*  
 \* @param req: 请求d  
 \* @param res: 返回  
 \* @return: null  
 \* @date: 2021/5/25 17:31  
 \* @description:截止当前，在前N个侵权站点，发现的侵权数量分布。  
 \*/  
export async function handleTortCountGroupByTortSite(req, res) {  
  
 console.time('handleTortCountGroupByTortSite');  
 let sqlRes = await getTortCountGroupByTortSite();  
 console.timeEnd('handleTortCountGroupByTortSite');  
 console.log('--------------------');  
 return sqlRes;  
}  
async function getTortCountGroupByTortSite() {  
 let TortCountGroupByTortSite = [];  
 let TortSiteInfo = {};  
 if(CONNECT == true){  
 let Res = await countGroupBy("tort\_info","site\_name");  
 let keys = Object.keys(Res);  
 for (let i = 0, n = keys.length, key; i < n; ++i) {  
 key = keys[i];  
 TortSiteInfo = {  
 "TortSite":key,  
 "TortCount":Res[key]  
 };  
 TortCountGroupByTortSite.push(TortSiteInfo);  
 }  
 if(keys.length<3){  
 \_PROTECT\_getTortCountGroupByTortSite(TortCountGroupByTortSite);  
 }  
 }  
 else{  
 \_PROTECT\_getTortCountGroupByTortSite(TortCountGroupByTortSite);  
 }  
 console.log(TortCountGroupByTortSite);  
 return TortCountGroupByTortSite;  
}  
function \_PROTECT\_getTortCountGroupByTortSite(TortCountGroupByTortSite) {  
 console.log("\_PROTECT\_getTortCountGroupByTortSite");  
 let TortSiteInfo = {};  
 TortSiteInfo = {  
 "TortSite":TORTSITE["3"],  
 "TortCount":0  
 };  
 TortCountGroupByTortSite.push(TortSiteInfo);  
 TortSiteInfo = {  
 "TortSite":TORTSITE["2"],  
 "TortCount":0  
 };  
 TortCountGroupByTortSite.push(TortSiteInfo);  
 TortSiteInfo = {  
 "TortSite":TORTSITE["1"],  
 "TortCount":0  
 };  
 TortCountGroupByTortSite.push(TortSiteInfo);  
}  
  
/\*  
 \* @param req: 请求  
 \* @param res: 返回  
 \* @return: null  
 \* @date: 2021/5/25 17:31  
 \* @description:截止当前，在前N个侵权站点，发现的侵权数量分布。  
 \*/  
export async function handleTortCountGroupByTortSiteEXchange(req, res) {  
  
 console.time('handleTortCountGroupByTortSiteEXchange');  
 let sqlRes = await getTortCountGroupByTortSiteEXchange();  
 console.timeEnd('handleTortCountGroupByTortSiteEXchange');  
 console.log('--------------------');  
 return sqlRes;  
}  
async function getTortCountGroupByTortSiteEXchange() {  
 let TortCountGroupByTortSiteEXchange = [];  
 let TortCountGroupByTortSite = [];  
 let TortSiteInfo = {};  
 let [TimeStampArray,MonthArray] = DateUtil.getMonthTimeStampArray();  
 let MonthGap = 1;  
 if(CONNECT == true){  
 let index = 0;  
 let endTimeStamp = TimeStampArray[index];  
 let startTimeStamp = TimeStampArray[(index + 1)];  
 let Res = await countGroupBy("tort\_info","site\_name", endTimeStamp , startTimeStamp, "monitor\_time");  
 let keys = Object.keys(Res);  
 for (let i = 0, n = keys.length, key; i < n; ++i) {  
 key = keys[i];  
 let MonthInfo = {  
 "TortSite":key,  
 "TortCount":Res[key],  
 "Month" : MonthArray[index + MonthGap],  
 };  
 TortCountGroupByTortSite.push(MonthInfo);  
 }  
 TortCountGroupByTortSiteEXchange.push(TortCountGroupByTortSite);  
 index = index + MonthGap;  
 for (; index < 12; index = index + MonthGap) {  
 let TortCountGroupByTortSite = [];  
 let endTimeStamp = TimeStampArray[index];  
 let startTimeStamp = TimeStampArray[(index + 1)];  
 let Res = await countGroupBy("tort\_info","site\_name", endTimeStamp ,startTimeStamp,"monitor\_time");  
 for (let i = 0, n = keys.length, key; i < n; ++i) {  
 key = keys[i];  
 if(Res[key]==null)Res[key]=0;  
 let MonthInfo = {  
 "TortSite":key,  
 "TortCount":Res[key],  
 "Month" : MonthArray[index + MonthGap],  
 };  
 TortCountGroupByTortSite.push(MonthInfo);  
 }  
 TortCountGroupByTortSiteEXchange.push(TortCountGroupByTortSite);  
 }  
 }  
 else{  
 for (let index = 0; index < 12; index = index + MonthGap) {  
 let TortCountGroupByTortSite = [];  
 TortSiteInfo = {  
 "TortSite": TORTSITE["3"],  
 "TortCount": 0,  
 "Month": MonthArray[index + MonthGap],  
 };  
 TortCountGroupByTortSite.push(TortSiteInfo);  
 TortSiteInfo = {  
 "TortSite": TORTSITE["2"],  
 "TortCount": 0,  
 "Month": MonthArray[index + MonthGap],  
 };  
 TortCountGroupByTortSite.push(TortSiteInfo);  
 TortSiteInfo = {  
 "TortSite": TORTSITE["1"],  
 "TortCount": 0,  
 "Month": MonthArray[index + MonthGap],  
 };  
 TortCountGroupByTortSite.push(TortSiteInfo);  
 TortCountGroupByTortSiteEXchange.push(TortCountGroupByTortSite);  
 }  
 }  
 console.log(TortCountGroupByTortSiteEXchange);  
 return TortCountGroupByTortSiteEXchange;  
}  
  
/\*  
 \* @param req: 请求  
 \* @param res: 返回  
 \* @return: null  
 \* @date: 2021/5/25 17:31  
 \* @description:截止当前，前N个侵权站点发现的侵权数量在不同作品类型下的分布。  
 \*/  
export async function handleTortCountGroupByTortSiteGroupByWorkType(req, res) {  
  
 console.time('handleTortCountGroupByTortSiteGroupByWorkType');  
 let sqlRes = await getTortGroupByTortSiteGroupByWorkType();  
 console.timeEnd('handleTortCountGroupByTortSiteGroupByWorkType');  
 console.log('--------------------');  
 return sqlRes;  
}  
async function getTortGroupByTortSiteGroupByWorkType() {  
 let TortCountGroupByWorkType = [];  
 let WorkTypeInfo = {};  
 let totalTortCount = 0;  
 if(CONNECT == true) {  
 let Res = await countGroupBy("tort\_info", "site\_name", null, null, null, 6);  
 let keys = Object.keys(Res);  
 let sqlRight = util.format("SELECT\n" +  
 "\tCOUNT(Type.work\_id) AS num, \n" +  
 "\tType.work\_type\n" +  
 "FROM\n" +  
 "\t(\n" +  
 "\t\tSELECT\n" +  
 "\t\t\twork\_info.work\_type, \n" +  
 "\t\t\ttort\_info.work\_id\n" +  
 "\t\tFROM\n" +  
 "\t\t\ttort\_info\n" +  
 "\t\t\tINNER JOIN\n" +  
 "\t\t\twork\_info\n" +  
 "\t\t\tON \n" +  
 "\t\t\t\ttort\_info.work\_id = work\_info.work\_id\n" +  
 "\t\tWHERE\n" +  
 "\t\t\ttort\_info.site\_name = \"%s\"OR\n" +  
 "\t\t\ttort\_info.site\_name = \"%s\"OR\n" +  
 "\t\t\ttort\_info.site\_name = \"%s\"\n" +  
 "\t) AS Type\n" +  
 "GROUP BY\n" +  
 "\twork\_type\n" +  
 "ORDER BY\n" +  
 "\tnum DESC\n" +  
 "LIMIT 6", keys[0], keys[1], keys[2]);  
 let sqlRes = await mysqlUtils.sql(c, sqlRight);  
 sqlRes.forEach(function (item) {  
 let WorkTypeInfo = {  
 "workType": WORKTYPE[item['work\_type']],  
 "TortCount": item['num'],  
 };  
 TortCountGroupByWorkType.push(WorkTypeInfo);  
 });  
 if (TortCountGroupByWorkType.length < 6) {  
 \_PROTECT\_getTortGroupByTortSiteGroupByWorkType(TortCountGroupByWorkType);  
 }  
 }  
 else{  
 \_PROTECT\_getTortGroupByTortSiteGroupByWorkType(TortCountGroupByWorkType);  
 }  
 console.log(TortCountGroupByWorkType);  
 return TortCountGroupByWorkType;  
}  
function \_PROTECT\_getTortGroupByTortSiteGroupByWorkType(TortCountGroupByWorkType) {  
 console.log("\_PROTECT\_getTortGroupByTortSiteGroupByWorkType");  
  
 for(let i = 1;i <= 6;i++){  
 let WorkTypeInfo = {  
 "workType":WORKTYPE[i],  
 "TortCount":0,  
 };  
 TortCountGroupByWorkType.push(WorkTypeInfo);  
 }  
}  
/\*  
 \* @param req: 请求  
 \* @param res: 返回  
 \* @return: null  
 \* @date: 2021/5/25 17:31  
 \* @description:截止当前，在不同作品类型下的侵权维权分布。  
 \*/  
export async function handleTort\_AND\_ClaimCountGroupByWorkType(req, res) {  
  
 console.time('handleTort\_AND\_ClaimCountGroupByWorkType');  
 let sqlRes = await getTortTort\_AND\_ClaimCountGroupByWorkType();  
 console.timeEnd('handleTort\_AND\_ClaimCountGroupByWorkType');  
 console.log('--------------------');  
 return sqlRes;  
}  
async function getTortTort\_AND\_ClaimCountGroupByWorkType() {  
 let TortCountGroupByWorkType = [];  
 let WorkTypeInfo = {};  
 let totalTortCount = 0;  
 if(CONNECT == true){  
 let Res = await countNum("tort\_info","sample\_id");  
 totalTortCount = Res['num'];  
 let sqlRight = util.format("SELECT\n" +  
 "\tType.work\_type, \n" +  
 "\tCOUNT(Type.work\_id) AS num\n" +  
 "FROM\n" +  
 "\t(\n" +  
 "\t\tSELECT DISTINCT\n" +  
 "\t\t\ttort\_info.work\_id, \n" +  
 "\t\t\twork\_info.work\_type\n" +  
 "\t\tFROM\n" +  
 "\t\t\ttort\_info\n" +  
 "\t\t\tINNER JOIN\n" +  
 "\t\t\twork\_info\n" +  
 "\t\t\tON \n" +  
 "\t\t\t\ttort\_info.work\_id = work\_info.work\_id\n" +  
 "\t) AS Type\n" +  
 "GROUP BY\n" +  
 "\tType.work\_type\n" +  
 "ORDER BY\n" +  
 "\tnum DESC\n" +  
 "LIMIT 6");  
 let sqlRes = await mysqlUtils.sql(***c***, sqlRight);  
 sqlRes.forEach(function(item){  
 let WorkTypeInfo = {  
 "workType":***WORKTYPE***[item['work\_type']],  
 "TortCount":item['num'],  
 "TotalTortCount": totalTortCount,  
 "ClaimCount": 0  
 };  
 TortCountGroupByWorkType.push(WorkTypeInfo);  
 });  
 if(TortCountGroupByWorkType.length<6){  
 \_PROTECT\_(TortCountGroupByWorkType);  
 }  
 }  
 else{  
 \_PROTECT\_(TortCountGroupByWorkType);  
 }  
 ***console***.log(TortCountGroupByWorkType);  
 return TortCountGroupByWorkType;  
 function \_PROTECT\_(TortCountGroupByWorkType){  
 for(let i = 1;i <= 6;i++){  
 let WorkTypeInfo = {  
 "workType":***WORKTYPE***[i],  
 "TortCount":0,  
 "TotalTortCount":0,  
 "ClaimCount":0  
 };  
 TortCountGroupByWorkType.push(WorkTypeInfo);  
 }  
 }  
}