大事件（back）项目笔记文档

登录功能

步骤：

1 对提交按钮进行处理

方式1：阻止默认的提交行为 方式2：修改为普通按钮

2 给按钮设置点击事件

3 检测两个输入框内容是否为空

4 如果内容检测通过，将内容发送给接口进行检测

5 接收响应内容

5.1 成功时跳转到index.html

5.2 失败时进行错误提示即可

```
    $('.input_sub').on('click', function (event) {

            event.preventDefault() //阻止表单默认提交事件
            var input_txt = $('.input_txt').val().trim()

            var input_pass = $('.input_pass').val().trim()

            if (!input_txt) {
                alert('用户名不能为空')
                return false
            }
            if (!input_pass) {
                alert('密码不能为空')
                return false
            }
            // alert(1)


            $.ajax({
                url: 'http://localhost:8000/admin/login',
                type: 'post',
                data: {
                    user_name: input_txt,
                    password: input_pass
                },
                success: function (res) {
                    console.log(res)
                    if (res.msg == '登录成功') {
```

```
                    localStorage.setItem('username',input_txt)

                    setTimeout(function () {


                        window.location.href = './index.html'
                    }, 1000);
                } else {
                    alert(res.msg)
                    $('.input_txt').val('')
                    $('.input_pass').val('')
                }
            },
            error: function (err) {
                //  alert(err)


            }
        })
    })
```

退出功能：

步骤：

1 点击退出按钮

2 设置确认是否退出

3 如果确定，请求接口进行退出操作(清空本地缓存的数据)

4 跳转到登录页面

```
//   1 点击退出按钮
$('#logout').on('click',function(event) {
   event.preventDefault()

    // 2 设置确认是否退出
     var flag=window.confirm('确定是否要退出')
     if(flag) {
        // 3 如果确定，请求接口进行退出操作(清空本地缓存的数据)
        $.ajax({
            url:'http://localhost:8000/admin/logout',
            type:'post',
            success:function(res) {
```

```
                console.log(res.msg)
                if(res.msg=="登出成功") {
                    //1    清空本地缓存
                    // 2   过1秒跳转到login页面
                   localStorage.removeItem('username')
                   setTimeout(function(){
                       location.href='./login.html'
                   },1000)
                }

            }
        })
    }



})
```

首页左侧用户基本信息展示：

步骤：

1 页面加载过程中直接请求用户信息

2 将响应数据展示在页面中即可

```
    // 判断登陆是否成功
    var  get_user_name=localStorage.getItem('username')
        if(!get_user_name) {
          location.href='./login.html'

        }

        $('#user_name').text(get_user_name)

        $.ajax({
             url:'http://localhost:8000/admin/getuser',
             success:function(res) {
                 console.log(res)
                 var data=res.data;
                 var  nick_name=data.nickname
                 var user_pic=data.user_pic

                 $('#person').prop('src',user_pic)
                 $('#nick_name').text(nick_name)

             }
        })
```

文章类别获取：

（ http://localhost:8000/admin/category_search ）

1 分类数据展示功能

- 发送请求，获取数据

- 使用模板引擎进行结构生成

- 设置模板

- 调用模板方法，将数据和模板结合得到要生成的结构字符串

- 生成到页面中即可

```
//html模板

    <script type="text/html" id='selCategoryTmp'>
        {{each data}}
        <tr>
            <td>{{$value.name}}</td>
            <td>{{$value.slug}}</td>
            <td class="text-center">
                <a
href="javascript:editTr({&amp;quot;id&amp;quot;:&amp;quot;2&amp;quot;,&amp;quot;slug&amp;quot;:&amp;quot;funny&amp;quot;,&amp;quot;name&amp;quot;:&amp;quot;奇趣事&amp;quot;});"
                    class="btn btn-info btn-xs" data-id={{$value.id}}>编辑</a>
                <a href="javascript:deleteTr( 2 );" class="btn btn-danger btn-xs" data-id=
{{$value.id}}>删除</a>
            </td>
        </tr>

        {{/each}}
    </script>




    // ?date='+new Date() //遇到服务器缓存的时候再发请求的后面加上当前时间
        $.ajax({
            url: 'http://localhost:8000/admin/category_search',
            success: function (res) {
                console.log(res)
                if (res.code == 200) {
```

```
                    var selCategory_html = template('selCategoryTmp', res)

                    console.log(selCategory_html)

                    $('#selCategory tr').eq(0).after(selCategory_html)

                }
            }
        })
```

2 分类数据新增功能

- 点击新增按钮，进行内容检测

- 填写完毕，发送请求

- 新增成功后，调用location.reload()刷新页面(iframe中的小区域)

```
        //新增分类

        $('#model_add').on('click', function () {

                event.preventDefault() //阻止表单默认提交事件

            //          1   点击新增按钮，进行内容检测
            // •          2    填写完毕，发送请求
            // •          3   新增成功后，调用location.reload()刷新页面(iframe中的小区域)


            // <button type="button" class="btn btn-primary" id="model_add">新增</button>
            //      <button type="button" class="btn btn-primary" id="model_edit">编辑
</button>


            var mycategory = $('.mycategory').val().trim()

            var mycategory_name = $('.mycategory_name').val().trim()

            if (!mycategory) {
                alert('用户名不能为空')
                return false
            }
            if (!mycategory_name) {
```

```
                alert('密码不能为空')
                return false
            }


            $.ajax({
                url: 'http://localhost:8000/admin/category_add',
                type: 'post',
                data: {
                    name: mycategory,
                    slug: mycategory_name
                },
                success: function (res) {
                    console.log(res)
                    if (res.msg == "新增成功") {
                        window.location.reload() //重新刷新页面
                    } else {
                        alert(res.msg)
                    }
                },

                complete: function () {
                    $('.mycategory').val('')
                    $('.mycategory_name').val('')
                }

            })

        })
```

3 分类数据编辑功能

- 编辑和新增使用的是同一个模态框

- 设置一个提交编辑按钮，用来进行编辑操作

- 进行操作效果处理：点击新增，将提交编辑按钮隐藏，点击编辑，将新增按钮隐藏

- 点击表格中的编辑按钮时，需要获取到数据的id

- 可以在创建结构时添加data-id属性保存

- 为了方便获取编辑按钮，添加了edit类名

- 在模态框中设置隐藏域，用来保存编辑的id

- 点击提交编辑，检测内容并发送请求

- 编辑成功后 重新刷新页面

```
    $('tbody').on('click', '.btn-info', function () {
            $('#addModal').modal('show');
            $('#addModal .modal-title').html('修改分类名称');
            $('#model_edit').show()
            $('#model_add').hide()


            var id = Number($(this).attr('data-id'))


            //隐藏新增按钮
            var this_tr = $(this).parents('tr')
            var td1 = this_tr.children().eq(0).text()
            var td2 = this_tr.children().eq(1).text()
            console.log(td1)
            console.log(td2)
            //赋值
            $('.mycategory').val(td1)
            $('.mycategory_name').val(td2)


            $('#model_edit').on('click', function () {
                $.ajax({
                    url: 'http://localhost:8000/admin/category_edit',
                    type: 'post',
                    data: {
                        id: id,
                        name: $('.mycategory').val(), //重新获取值
                        slug: $('.mycategory_name').val() //重新获取值
                    },
                    success: function (res) {
                        console.log(res)
                        if (res.code == 200) {
                            location.reload()
                        }
                    }
                })
            })

        })
```

4 分类数据删除功能

- 点击删除按钮，获取data-id

- 将data-id发送给接口进行删除操作

具体步骤：

1 使用事件委托给删除按钮添加点击事件

2 获取data-id属性

3 调用接口发送id

```
        $('tbody').on('click', '.btn-danger', function () {
            //   alert($(this).attr('data-id'))
            var id = Number($(this).attr('data-id'))
            var flag = window.confirm('是否真的要删除')
            if (!flag) {
                return false;
            }
            $.ajax({
                url: 'http://localhost:8000/admin/category_delete',
                type: 'post',
                data: {
                    id: id
                },
                success: function (res) {
                    console.log(res)
                    if (res.code == 200) {

                        location.reload()
                    }
                }
            })

        })
```

关闭弹框清空内容

```
    $('#model_shutoff').click(function () {
            $('.mycategory').val('')
            $('.mycategory_name').val('')
        });
```

切换编辑与新增

```
$('.text-center') .on('click',function() {
        $('#model_edit').hide()
        $('#model_add').show()
        $('.modal-title').html('新增分类名称');
```

文章信息展示以及分页功能：

功能步骤: 1 文章信息展示 - 请求接口，获取数据 - 设置模板，生成结构即可 2 文章信息分页展示 3 文章信息筛选

```
// 1 调用函数，进行初始结构创建
getArticle();
// 2 文章信息分页展示(设置位置，在getArticle中)
//    - 使用jQuery插件，引入文件
//    - 调用插件方法，设置分页结构
//      - 总页数是必须的，需要由接口响应给我们
//        - 必须在第一次请求到数据后进行操作
//    - 点击页码，将页码发送给服务端获取新数据
//    - 根据数据进行表格展示
var page = 1; // 设置全局变量，用来记录当前展示的页码


var lastTotalPage = 0; // 设置全局变量，用于保存当前总页数
var $list = $('#list'); // 提前保存一下分页的父元素
// 3 设置文章数据筛选功能
//  3.1 请求分类数据，进行下拉菜单结构创建
//    - 在分类操作页面封装了article.getCate()用来请求分类数据
article.getCate({
  success: function (res) {
    // - 通过模板引擎进行结构创建
    $('#selCategory').append(template('category', res));
  }
});
//  3.2 筛选按钮点击操作
$('#btnSearch').on('click', function () {
  // 调用之前的函数，但是需要对请求参数进行处理
  getArticle();
  // 为了避免筛选时，两次结果的总页数相同，页码不会更新
  //   - 自己手动触发首页按钮点击事件即可，first类名是插件自己添加的
  $('.first').click();
});
// 用来进行请求发送的函数
function getArticle() {
  // 1 请求数据并进行结构创建
  $.ajax({
    url: 'http://localhost:8000/admin/search',
    // 当前接口有很多参数，但是首页的基本数据获取无需传入参数
```

```
      data: {
        page: page, // 分页参数
        type: $('#selCategory').val(), // 分类筛选参数
        state: $('#selStatus').val() // 状态筛选参数
      },
      success: function (res) {
        // 检测响应状态
        if (res.code === 200) {
          // 通过模板生成结构即可
          $('tbody').html(template('article', res));
          // 获取到的res中存在totalPage的属性，表示本次请求数据的总页数
          var totalPage = res.totalPage;

          // 需要在重新生成分页结构前进行检测，如果总页数与上次相同，没必要重新创建了
          if (totalPage === lastTotalPage) {
            return;
          }
          // 每次重置分页结构时，记录上一次的总页数
          lastTotalPage = totalPage;

          // 如果进行了筛选，需要将旧的分页结构删除（文档中提供的删除方式）
          $list.twbsPagination('destroy');
          // 进行分页结构的创建操作
          $list.twbsPagination({
            totalPages: totalPage,
            visiblePages: 8,
            first: '首页',
            last: '尾页',
            prev: '上一页',
            next: '下一页',
            onPageClick: function (e, currentPage) {
              // 点击页码按钮后，将页码发送给接口，请求新一页的数据
              page = currentPage;
              getArticle();
            }
          });
        }
      }
    });
  }
```

用户的信息获取与展示

1 请求用户的详细信息

- 给表单中的元素设置与相应数据相同的id，方便获取

2 将信息展示在页面中即可

```
        $.ajax({
            url: 'http://localhost:8000/admin/userinfo_get',
            //  data:{},
            success: function (res) {
                console.log(res)

                var user_pic = res.data.user_pic

                $.each(res.data, function (key, val) {
                    console.log(key)
                    $('.' + key).val(val)
                })

                $('.user_pic').prop('src', user_pic)

            }


        })
```

用户的信息编辑功能

- 1 检测是否完整填写表单

- 输入框检测val()，文件域检测是否选取文件

- 2 给表单中的元素设置name属性，否则无法提交

- 进行提交按钮处理

- 3 提交到服务端进行编辑

- 4 图片的本地预览

```
// 1 点击按钮，检测表单是否填写完毕
$('#btn').on('click', function () {
  // 1.1 检测内容
  //    - 检测文件域是否选择文件，
  //    使用DOM对象的files属性，进行length检测
  if (
    $('#username').val().trim() === '' ||
    $('#nickname').val().trim() === '' ||
    $('#email').val().trim() === '' ||
    $('#password').val().trim() === '' ||
    $('#user_pic')[0].files.length === 0
  ) {
    alert('内容没有填写完毕');
    return; // 阻止后续的请求发送
  }

  // 1.2 将数据发送给服务端处理（使用FormData即可）
  //    - 传入参数必须为DOM对象形式的form标签
  var fd = new FormData($('form')[0]);

  // 1.3 使用jQuery的ajax发送
  $.ajax({
    type: 'POST',
    url: 'http://localhost:8000/admin/userinfo_edit',
    data: fd,
    contentType: false, // 不需要jQuery设置内容类型
    processData: false, // 不需要jQuery进行内容处理
    success: function (res) {
      if (res.code === 200) {
        // 1.4 让当前user.html的iframe所在的index.html跳转到login.html
        //    - window.parent用来获取父页面的window对象，也就是index.html页面的window
        // window.location.href = './login.html'; // 当前小窗口部分跳转，不对
        parent.location.href = './login.html'; // 父页面跳转，是正确的效果
      }
    }
  });

});
```

图片本地预览

1 使用change事件监测用户的文件选择操作

2 通过 URL.createObjectURL()进行本地图片地址获取

- URL是window对象的属性

- 用户在本地选择的图片地址我们不可能提前知道

- 使用方式：

URL.createObjectURL(文件域的files中的文件信息)

- 返回值是浏览器自动生成的临时图片地址，可以设置在src中查看

```
 // 1 给文件域设置change事件
    $('#user_pic').on('change', function () {
      // console.log($(this)[0].files); // 查看文件域选择的文件信息
      // console.log(URL.createObjectURL($(this)[0].files[0])); // 将某个文件的信息传入到方法中
      var tempSrc = URL.createObjectURL($(this)[0].files[0]);
      $('#avatar').prop('src', tempSrc); // 将方法生成的临时图片地址设置给img展示即可
    });
```

日期控件的的基本使用

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>Document</title>
  <link rel="stylesheet" href="./web_back/js/jedate/css/jedate.css">
</head>

<body>
  <input type="text" id="test" readonly>
  <button id="btn">显示日期</button>
  <script src="web_back/js/jedate/js/jedate.js"></script>
  <script>
    // 一个原生js编写的日期控件，地址： http://www.jemui.com/uidoc/jedate.html


    // 设置btn的事件，点击btn后弹出日期控件效果
    document.querySelector('#btn').onclick = function () {
      jeDate("#test", {
        format: "YYYY-MM-DD",
        isTime: false,
        trigger: false, // 不使用内置的事件功能，自己进行设置
        minDate: "2014-09-19 00:00:00"
      });
    };

  </script>
</body>

</html>
```

分页功能基本使用:

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>Document</title>
  <link rel="stylesheet" href="./web_back/js/bootstrap/css/bootstrap.min.css">
</head>

<body>
  <!-- 用来设置分页的标签结构 -->
  <ul id="list"></ul>

  <!-- 引入jQuery和插件文件，注意引入顺序 -->
  <script src="./web_back/js/jquery-1.12.4.min.js"></script>
  <script src="./web_back/js/jquery.twbsPagination.js"></script>

  <script>
    // 根据文档，设置分页展示功能
    $('#list').twbsPagination({
      totalPages: 20,
      visiblePages: 7, // 设置显示的页码个数为 visiblePages
      first: '首页',
      last: '尾页',
      prev: '上一页',
      next: '下一页',
      onPageClick: function (e, page) {
        console.log(e)
        console.log(page); // 获取当前点击的页码
      }
    });



  </script>
</body>

</html>
```

iframe标签的基本使用

```html
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <style>
    html,
    body {
      height: 100%;
      margin: 0;
      padding: 0;
    }

    iframe {
      width: 100%;
      height: 100%;
    }
  </style>
  <title>Document</title>
</head>

<body>
  <button id="btn">修改iframe的src切换展示的页面</button>
  <iframe id="iframe" src="http://www.baidu.com" frameborder="0"></iframe>
  <script>
    document.getElementById('btn').onclick = function () {
      document.getElementById('iframe').src = 'http://www.taobao.com';
    }



  </script>
</body>

</html>
```