1.AJAX

1.什么是AJAX

Asynchronous Javascript And Xml
异步的 JS 和 xml(*EX*tensible *M*arkup *L*anguage)

通过 JS 异步的向服务器发送请 求并接收响应数据

同步访问：
当客户端向服务器发送请求时，服务器在处理的过程中，浏览器只能等待，效率较低

异步访问：
当客户端向服务器发送请求时，服务器在处理的过程中，客户端可以做其他的操作，不需要一直等待

AJAX优点：

1.异步访问

2.局部刷新

使用场合：

1.搜索建议

2.表单验证

3.前后端分离

2.AJAX核心对象 - 异步对象(XMLHttpRequest)

1.什么是XMLHttpRequest [简称为 xhr]

称为 "异步对象"，代替浏览器向服务器发送异步的请求并接收响应

[xhr 是由JS来提供的]

2.创建 异步对象 (xhr)

1.IE7+,Chrome,Firefox,Safari,Opera) -> 调用 XMLHttpRequest 生成 xhr对象

2.IE低版本浏览器中(IE6以及以下) -> 调用 ActiveXObject() 生成xhr

```
<script>
    if(window.XMLHttpRequest){
        //支持 XMLHttpRequest
        var xhr = new XMLHttpRequest();
    }else{
        //不支持XMLHttpRequest,使用 ActiveXObject 创建异步对象
        var xhr = new ActiveXObject("Microsoft.XMLHTTP");
    }
</script>
```

3.xhr 的成员

1.方法 - open()

作用：创建请求

语法：open(method,url,asyn)

参数：

method:请求方式，取值'GET' 或 'POST'

url:请求地址，字符串

asyn:是否采用异步的方式 - true:异步 / false:同步

ex: xhr.open('GET','/server',true);

2.方法 - send()

作用：通知xhr向服务器端发送请求

语法：send(body)

参数：

GET请求：body的值为null -> send(null)

POST请求：body的值为请求数据 -> send("请求数据")

3.属性 - readyState

作用：xhr状态，通过不同的xhr状态来表示xhr与服务器的交互情况

由0-4共5个值来表示5个不同的状态

状态	说明
0	代理被创建，但尚未调用 open() 方法。
1	`open()` 方法已经被调用。
2	`send()` 方法已经被调用，响应头也已经被接收
3	下载中； `responseText` 属性已经包含部分数据。
4	下载操作已完成

4.属性 - responseText

作用：响应数据

5.属性 - status

作用：服务器端的响应状态码

状态吗	说明
200	表示服务器正确处理所有的请求以及给出响应
404	请求资源不存在
500	服务器内部错误

6.事件 - onreadystatechange

作用：每当xhr的readyState发生改变的时候都要触发的操作；

也称作回调函数；当readyState的值为4且status值为200的时候，才可以获取响应数据

3.AJAX的操作步骤

1.GET请求

```
//1.创建xhr请求
var xhr = createXhr();
//2.创建请求 - open()
xhr.open('GET',url,asyn[true|false])
//3.设置回调函数 - onreadystatechange
xhr.onreadystatechange = function(){
    if(xhr.readyState == 4 && xhr.status == 200){
        //接收响应
        xhr.responseText;

        JSON对象=JSON.parse( xhr.responseText)
    }
}
//4.发送请求
xhr.send(null);

//注意：若含有请求参数 - URL后拼接 查询字符串 QueryString
//ex: xhr.open('get','/url?key=value&key=value',asyn)
```

练习：注册框的 用户名检查

2.POST请求

```
//1.创建xhr请求
var xhr = createXhr();
//2.创建请求 - open()
xhr.open('post',url,asyn[true|false])
//3.设置回调函数 - onreadystatechange
xhr.onreadystatechange = function(){
    if(xhr.readyState == 4 && xhr.status == 200){
        //接收响应
        xhr.responseText;
    }
}
//4设置Content-Type;
//默认ajax post的Content-Type为 "text/plain;charset=utf-8"
xhr.setRequestHeader('Content-Type','application/x-www-form-urlencoded');
//5.发送请求
xhr.send('请求数据');
//请求数据同查询字符串 "uname=guoxiaonao&age=18"
```

注意：django中post需要传递csrf_token,否则触发响应码403，拒绝访问；

获取csrf_token方法如下

```
var csrf=$("[name='csrfmiddlewaretoken']").val();
#获取后，将token放在post body数据中一并提交
```

2.jquery对 ajax 的支持

$.ajax({})

```
参数对象中的属性：
    1.url ： 字符串，表示异步请求的地址
    2.type ： 字符串，请求方式，GET 或 POST
    3.data ： 传递到服务器端的参数
        可以是字符串 ："name=sf.zh&age=18"
        也可以是js对象：
            {
                name:"sf.zh",
                age:18
            }
    4.dataType ： 字符串，响应回来的数据的格式
        1.'html'
        2.'xml'
        3.'text'
        4.'script'
        5.'json'   ： jq将json字符串转化成json对象
        6.'jsonp' ： 有关跨域的响应格式
    5.success:回调函数，请求和响应成功时回来执行的操作
    6.error ： 回调函数，请求或响应失败时回来执行的操作
    7.beforeSend ： 回调函数，发送ajax请求之前执行的操作，如果return false，则终止请求
    8.contentType ： 当有请求体有数据提交时，标明提交方式，默认值为'application/x-www-
form-urlencoded; charset=UTF-8'
```

3.JSON

1.JSON介绍

JSON:JavaScript Object Notation

定义：是一种轻量级的数据交换格式。JS的一个语法子集；采用完全独立于编程语言的文本格式来存储和表示数据。简洁和清晰的层次结构使得 JSON 成为理想的数据交换语言。 易于人阅读和编写，同时也易于机器解析和生成，并有效地提升网络传输效率。

场景：在ajax中，允许将 复杂格式的响应数据 构建成 JSON的格式再进行响应

2.JSON表现

1.JSON表示单个对象

1.使用 {} 表示单个对象

2.在 {} 中使用 key:value 的形式来表示属性(数据)

3.Key必须要用 " " 引起来

4.value如果是字符串的话，也需要用" "引起来

```
var obj = {
        "name":"王老师",
        "age" : 30,
        "gender" : "Unknown"
}
```

2.JSON表示一个数组

1.使用 [] 表示一个数组

2.数组中允许包含若干JSON对象 或 字符串

1.使用JSON数组表示若干字符串

```
var arr = ["王伟超","王夫人","王小超"];
```

2.使用JSON数组表示若干对象

```
var arr = [
    {
        "name":"王老师",
        "age":30,
        "gender":"男"
                        },
    {
        "name":"王夫人",
        "age":28,
        "gender":"男"
                        }
    ];
```

3.使用 jq 的 each() 迭代数组

回顾 JS中遍历数组

```
var a = [{"name":"guoxiaonao", "age": 18 }, {"name":"guoxiaonao2", "age": 22}];

for (var i = 0 ; i < a.length ; i++ ){
    var obj = a[i];
    console.log('name is ' + obj.name);
    console.log('age is '+ obj.age);
}
```

1.$arr.each();

$arr : jQuery中的数组

```
        //语法:


       var json_arr = []
       $(json_arr).each()


        $arr.each(function(index,obj){
            index:遍历出来的元素的下标
            obj:遍历出来的元素
        });
```

2.$.each()

```
        //语法:
        $.each(arr,function(index,obj){});
        arr : js 中的普通数组
```

4.后台处理JSON

在后台查询出数据再转换为JSON格式的字符串，再响应给前端

1.后台先获取数据

类型允许为：元组|列表|字典

2.在后台将数据转换为符合JSON格式的字符串

3.在后台将JSON格式的字符串进行响应

5.Python中的JSON处理

```
import json
#序列化 - python对象变为json字符串
jsonStr = json.dumps(元组|列表|字典)
#反序列化 - json字符串变为python对象
py_obj = json.loads(jsonStr)
```

Django中的JSON处理

```
#方法1 使用Django中提供的序列化类来完成QuerySet到JSON字符串的转换
from django.core import serializers
json_str = serializers.serialize('json',QuerySet)
return HttpResponse(json_str, content_type='application/json')

#方法2  推荐
d = {'a': 1}
#如果参数非字典结构，会报TypeError; 如果需要序列化非字典结构的对象，需要添加  safe=False 参数
#注意，JsonResponse会将 响应头 ct 由 html 改为 json
return JsonResponse(d)
```

6.前端中的JSON处理

```
#序列化
JSON字符串=JSON.stringify(JSON对象)

#反序列化
JSON对象=JSON.parse(JSON字符串)
```

练习：

1，页面点击摁钮 ， 发送ajax 请求至后端获取用数据

2，将后端返回的用户数据显示至页面

4,跨域

1，什么是跨域

跨域：非同源的网页，相互发送xhr请求的过程，就是跨域

```
浏览器的同源策略：
同源：多个地址中，相同协议，相同域名，相同端口被视为是"同源"
在HTTP中，必须是同源地址才能互相发送请求，非同源拒绝请求(<script src>和<img src>除外)。

http://www.tedu.cn/a.html
http://www.tedu.cn/b.html
以上地址是 "同源"

http://www.tedu.cn/a.html
https://www.tedu.cn/b.html
由于 协议不同 ， 所以不是"同源"

http://localhost/a.html
http://127.0.0.1/a.html
由于 域名不同 ， 所以不是"同源"

http://www.tedu.cn:80/a.html
http://www.tedu.cn:8080/b.html
由于端口不同 ， 所以不是"同源"
```

2，解决方案

通过 向服务器资源发送请求
由服务器资源指定前端页面的哪个js方法来执行响应的数据

3, jquery 的跨域

```
jsonp - json with padding
用户传递一个callback参数给服务端，然后服务端返回数据时会将这个callback参数作为函数名来包裹住JSON数据

ex:
    当前地址： http://127.0.0.1:8000/index
    欲访问地址： http://localhost:8000/data?callback=xxx
```

```
    $.ajax({
        url:'xxx',
        type:'GET',
        dataType:'jsonp',//指定为跨域访问
        jsonp:'callback',//定义了callback的参数名，以便获取callback传递过去的函数名
        jsonpCallback:'xxx' //定义jsonp的回调函数名
    });




    $.ajax({
        url:'xxx',
        type:'GET',
        dataType:'jsonp',//指定为跨域访问
        success: function(data){



        }
    });
```

ajax排错流程

```
1，后台写完之后，优先测试一下， 排除404/500

2，控制器中的js报错 + 绑定事件 准？

3，看调试工具~ 深入看
```