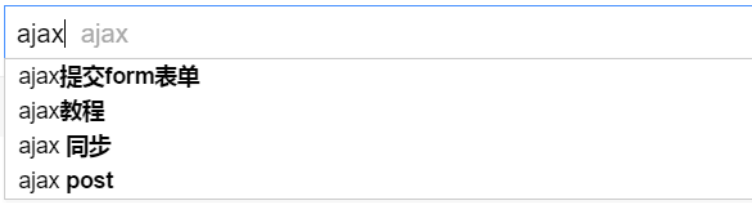
ajax入门

在没有ajax之前，前端与后台传数据都是靠表单传输，使用表单的方法传输数据有一个比较大的问题就是每次提交数据都会刷新页面，用户体验很不友好，解决方案有的是采用iframe，表单放在iframe中，不用刷新母页面，有的是在js里面重定位，这些方法都比较繁琐。所以人们发明了ajax这种异步通信方式。

ajax的应用

首先需要说明一点的是ajax是异步通信的，客户端向服务器请求一条数据后就不管了，继续执行下面的代码。接着是服务器，服务器那边接收到请求后就开始处理客户端的请求，处理完毕后再将数据发给客户端。然后客户端这时才继续执行相应的函数。

利用异步通信这一点，我们可以实现不用刷新页面就实现页面更新。利用ajax，我们可以实现很多有趣的功能。比如百度搜索的提示。当用户在搜索框输入字符时，下方会出现提示语。



百度的ajax

按下f12，选择network，会发现其实每次我们输入一个字符，百度的服务器那边就会通过ajax发送一条数据过来，这些数据就是下方的提示语。

如何使用ajax

这是一个简单的验证登录的js代码

```
        var username=document.getElementById("name").value;
                var password=document.getElementById("pwd").value;
                var xhr=null;
                xhr=new XMLHttpRequest();//新建一个XMLHttpRequest对象
                xhr.open("get",'./check.php',true);//初始化请求
                xhr.onreadystatechange=function () {//定义数据返回后的回调函数
                    if(xhr.readyState==4){
                        if(xhr.status===200){
                            var data=xhr.responseText;
                            if(data==1){
                                result.innerText="登录成功";
                            }
                            else if(data===2){
                                result.innerText="登录失败";
                            }
                        }
                    }
                };
                xhr.send(null);//xhr.send(null);

            };
```

ajax通信的步骤如下：

1. 新建一个XMLHttpRequest对象。
2. open方法表示初始化请求，此时并没有发送。
3. 定义数据返回后的回调函数，里面的代码在readystatechange值改变的时候执行。
4. 发送请求。

下面对上面提到的方法，参数做一个介绍。

open方法

用途：初始化请求。

参数一：请求方式，get或者post。

参数二：请求路径，数据被传输的目的地地址。

参数三：true：异步请求。false:同步请求。

Get请求

- 将表单数据以名称/值对的形式附加到 URL 中
- URL 的长度是有限的（大约 3000 字符）
- 绝不要使用 GET 来发送敏感数据！（在 URL 中是可见的）
- 对于用户希望加入书签的表单提交很有用
- GET 更适用于非安全数据，比如在 Google 中查询字符串

通过get的方式请求数据，那么对于send()方法，参数为null就可以。

get请求中如果有中文，可能会报错。所以，我们一般会对参数重新编码。

```
var url="open.php?username="+encodeURI(username)+"&password="+password;
```

ajax使用get方式是如何传输数据?

get方式的参数是放在url里面传输的，所以我们只要把参数放到url后面，拼接url字符串就可以了。

```
var url = './check.php?username='+username+"&password="+password;
```

Post请求

- 将表单数据附加到 HTTP 请求的 body 内（数据不显示在 URL 中）
- 没有长度限制
- 通过 POST 提交的表单不能加入书签

post请求传输数据的时候，数据是作为send方法的参数传输的。

```
var param = 'username='+username+'&password='+password;
xhr.send(param);
```

一般数据接收方，也就是后台对于post请求，会默认其数据类型是表单数据类型，所以需要我们设置一下post请求的数据格式，否则我们提交的数据会因为编码格式不对，导致后台取不到我们提交的数据。

```
xhr.setRequestHeader("Content-Type","application/x-www-form-urlencoded; charset=gb2312");
```

readyState

readyState只有5个值{0，1，2，3，4}，只读不能写。

0：XMLHttpRequest对象创建完成。

1：XMLHttpRequest对象初始化完成。

2：请求已经发送。

3：服务器已经返回了数据（但是还没有被解析，可能只一段http报文）。

4：数据解析已经完成。

status

status状态码，来自于http协议。详细的状态码以及意义可以参考http协议。推荐一本讲http协议的书籍，《图解http》，讲的比较简单，通熟易懂。

比较常见的：

200：成功

404：页面不存在

503：页面存在，但有语法错误

ResponseText和ResponseXml

数据返回的格式。

ResponseText表示返回的是一个字符串，也就是一个JSON。对象，和localstroge有点像。不管文件内容是什么都直接返回字符串。

ResponseXml表示返回的是一个xml格式的文档，如果返回的数据中有不符合xml格式的，则不会返回。比如文档内容如下：

```
<book>
        <name>红楼梦</name>
        <category>文学</category>
        <desc>宝哥哥与林妹妹的爱情史</desc>
    </book>
    helloworld
```

helloworld没有被标签闭合，不符合xml格式。

var data1=xhr.responseXML;打印data1会发现结果中并没有helloworld，因为它不符合xml格式，所以不会被传输过来。

url中的参数传递格式

url是一个地址，当我们需要传递参数的时候，首先需要在url后添加一个?号，然后后面跟上我们的参数。比如username="哈士奇",参数之间要用&号连接。比如username="哈士奇"&ppassword="我爱萨摩耶"。完整的例子如下：http://www.whutyzy.cc/index.php?username=哈士奇&ppassword=我爱萨摩耶。

jquery中的ajax

每次发送请求，都要写上这么一段代码：

```
var xhr=null;
xhr=new XMLHttpRequest();
xhr.open("get",url,true);
xhr.send(null);
xhr.onreadystatechange=function () {
    if(xhr.readyState===4){
        if(xhr.status===200){
            dosomething();
        }
    }
};
```

其中大部分的代码都一样，所以我们完全可以抽象成一个方法，所幸jquery方法以及帮我们封装好了一系列的ajax方法，我们只要直接调用就可以了。

使用jquery发送ajax请求的话，只要一行代码就可以了。

```
$.ajax({url:"test.js",dataType:"json"});
```

更多详细的信息请参考jquery-ajax教程。

jsonp

在讲jsonp是什么之前，我们先看下这么一个情况，我们想写一个天气预报的实时页面，数据肯定是从其他网站来的，比如url:"http://tq.360.cn/api/weatherquery/querys?app=tq360&code=$code&_jsonp=renderData&_=1429595284544"这个接口就提供了我们需要的天气数据。所以我们只需要使用ajax发送请求到这个网站上，再接收天气数据在页面上渲染就可以了。但是当我们写好代码运行

的时候会报错，提示No 'Access-Control-Allow-Origin' header is present on the requested resource. Origin 'http://localhost:63342' is therefore not allowed access.
大概意思是无法接收请求，原因是发生了跨域，也就是我们访问了其他域名，出于安全原因，我们不会接收其他域名发送过来的文件数据。对于这个跨域问题，有不同的解决方案，我们使用的方案是jsonp。
jsonp到底是什么？不要着急，继续往下看。
虽然浏览器不让我们接收其他网站的数据，但是允许我们加载其他网站上的script文件。比如比较常见的引入jquery：

```
<script src="https://cdn.bootcss.com/jquery/3.2.1/jquery.js"></script>
```

所以我们完全可以通过script的方式来加载数据。

```
<body>
<script src="https://cdn.bootcss.com/我爱哈士奇.php"></script>
</body>
```

假设这个我爱哈士奇.php的内容如下：

```
echo "helloworld";//输出helloworld
```

那么页面中<script src="https://cdn.bootcss.com/我爱哈士奇.php"></script>加载完后会返回helloworld。在页面中就是

```
<body>
<script>
helloworld
</script>
</body>
```

通过这样的方式，我们成功的获取了其他网站上的数据。
现在回到jsonp是什么这个问题上，jsonp就是值通过加载script文件这种方式解决跨域问题的一种方案（有问题，不过暂时这么理解）。但是实际上jsonp比上面这个例子稍微复杂一点。首先我们看一下html文件：

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>jsonp</title>
</head>
<body>
    <script type="text/javascript">
    function cb(data){
        console.log(data);
    }
    </script>
    <script type="text/javascript" src="https://cdn.bootcss.com/我爱哈士奇.php?_jsonp=cb">
</script>
</body>
</html>
```

在主体下面我们引入了一个我爱哈士奇的php文件，这个php文件内容如下：

```
<?php
$callback = $_GET['_jsonp'];
$arr = array("大金毛","哈士奇","短腿柯基");
echo $callback."(".json_encode($arr).")";
```

?>

这段php代码用js写的话代码如下

```
var callback=GET['_jsonp'];
var arr="哈士奇";
console.log(callback+"("+arr+")");
```

大概意思是获取url传过来的参数_jsonp的值并赋值给callback，然后再拼接字符串，拼成callback(arr)。然后再将拼好的字符串加载到页面上。

最后页面中js代码段内容如下

```
function cb(data){
        console.log(data);
}
cb("哈士奇");
```

这一段绕来绕去做了个什么？其实就是将传回来的数据放到之前定义好的函数中执行。因为后台的获取这个函数名是根据url传过来的参数取出来的，所以这个参数名_jsonp最好统一，方便后台程序员编写接口。

到这里，我们再看下jsonp的定义与背景：

1、一个众所周知的问题，Ajax直接请求普通文件存在跨域无权限访问的问题，甭管你是静态页面、动态网页、web服务、WCF，只要是跨域请求，一律不准；

2、不过我们又发现，Web页面上调用js文件时则不受是否跨域的影响（不仅如此，我们还发现凡是拥有”src”这个属性的标签都拥有跨域的能力，比如<script>、<img>、<iframe>；

3、于是可以判断，当前阶段如果想通过纯web端（ActiveX控件、服务端代理、属于未来的HTML5之Websocket等方式不算）跨域访问数据就只有一种可能，那就是在远程服务器上设法把数据装进js格式的文件里，供客户端调用和进一步处理；

4、恰巧我们已经知道有一种叫做JSON的纯字符数据格式可以简洁的描述复杂数据，更妙的是JSON还被js原生支持，所以在客户端几乎可以随心所欲的处理这种格式的数据；

5、这样子解决方案就呼之欲出了，web客户端通过与调用脚本一模一样的方式，来调用跨域服务器上动态生成的js格式文件（一般以JSON为后缀），显而易见，服务器之所以要动态生成JSON文件，目的就在于把客户端需要的数据装入进去。

6、客户端在对JSON文件调用成功之后，也就获得了自己所需的数据，剩下的就是按照自己需求进行处理和展现了，这种获取远程数据的方式看起来非常像AJAX，但其实并不一样。

7、为了便于客户端使用数据，逐渐形成了一种非正式传输协议，人们把它称作JSONP，该协议的一个要点就是允许用户传递一个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，这样客户端就可以随意定制自己的函数来自动处理返回数据了。

当然，jquery中也提供了封装好的jsonp方法。最后再提一句，一般不会选择在页面上直接引入<script>这种方式，而是在js中动态生成<script>节点。