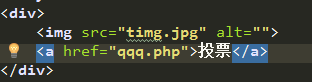
### 远古时代的方式

#### Header(‘Http/1.1 204 No Content’);

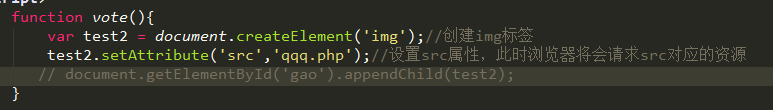
表示响应执行成功，但没有数据返回，浏览器不用刷新，不用导向新页面。



<?php  
*touch*('file.txt');  
$cont=*file\_get\_contents*('file.txt');  
$cont+=1;  
*file\_put\_contents*('file.txt',$cont);  
*header*('http/1.1 204 no content');

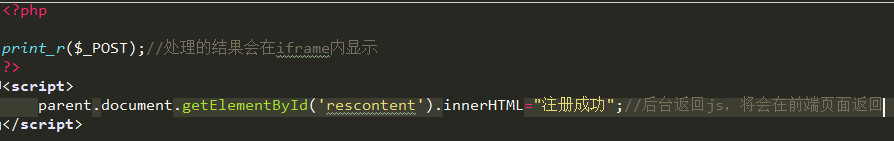
当浏览器收到204时，页面不做跳转

#### 利用图片加载特性



#### 利用iframe特性





使用from表单的target属性，将PHP返回的结果显示在iframe中，后将iframe隐藏掉即可。之后可以返回js代码来实现操作反馈。

### $.get()

$.get(‘action.php’,{“id”:”1”},function(res){

//此时，ajax返回的res是个json对象，可直接使用对象的属性，res.status等；

但若后面不跟‘json’，返回的就是json字符串的，此时必须先

res = JSON.parse(res), 【JSON要大写，小写没用】将json字符串转成json对象才可使用。

},’json’);

### 原生的Ajax

#### XMLHttpRequest对象(简称XHR对象)

|  |
| --- |
|  |

#### open()

open(‘get/post’,url,asyc)

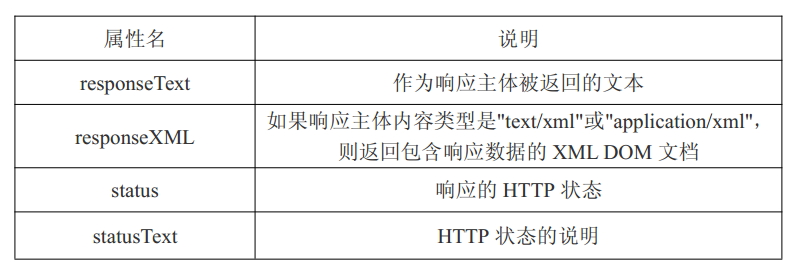
open()方法并不会真正发送请求，而只是启动一个请求以备发送。

#### send([data])

发送请求。

接受一个参数，作为请求主体发送的数据，如果不需要，必须填null，执行send()后，请求就会发送到服务器上。

当请求发送到服务器端， 收到响应后， 响应的数据会自动填充 XHR对象的属性。共四个属性：



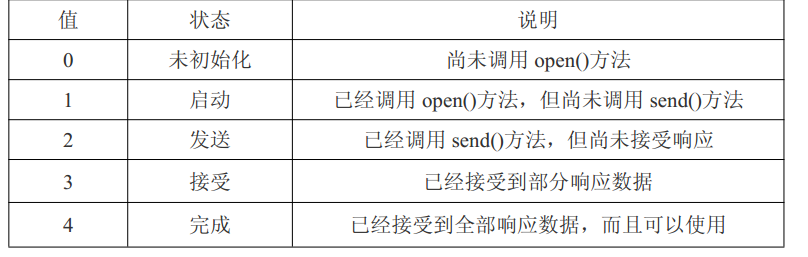
接受响应之后， 第一步检查 status属性， 以确定响应已经成功返回。 一般而已 HTTP状态代码为 200作为成功的标志。

同步：

|  |
| --- |
| document.onclick=function(){  var xhr = new XMLHttpRequest();  xhr.open('get','ajax.php?x='+Math.random(),false);  xhr.send(null);  if(xhr.status==200){  alert(xhr.responseText);  }else{  alert(xhr.status+':'+xhr.statusText);  }  } |

异步：

使用异步调用的时候，需要先触发readystatechange事件，然后检测readyState属性即可。这个属性有五个值：



|  |
| --- |
| *document*.onclick=function(){  var xhr = new XMLHttpRequest();  xhr.onreadystatechange = function(){  if(xhr.readyState==4){  if(xhr.status==200){  alert(xhr.responseText);  }else{  //错误时  alert(xhr.status+':'+xhr.statusText);  }  }  };  xhr.open('get','ajax.php?x='+*Math*.random(),true);  xhr.send(null);  } |

#### 请求头和响应头

请求头：客户端发向服务器的头部信息，可以设置，不能获取

响应头：服务器返回的头部信息，只能获取，不能设置

//getResponseHeader()获取单个响应头信息：

xhr.getResponseHeader('Content-Type')

//使用 getAllResponseHeaders()获取整个响应头信息

xhr.getAllResponseHeaders()

//使用 setRequestHeader()设置单个请求头信息

xhr.setRequestHeader('MyHeader', 'Lee'); //放在 open方法之后， send方法之前

#### get/post

get：

xhr.open(‘get’,url+’?’+param,true) 参数用 encodeURIComponent()进行编码处理

post:

post请求一般用来提交表单，但是和web表单提交是不同的，需要用XHR来模仿表单提交：

xhr.open(‘post’,url,true);

//模拟表单提交

xhr.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');

xhr.send(data);