Ajax优点：

1. 局部刷新，增强用户体验。
2. 异步服务器通信
3. 前端和后端负载均衡。

Ajax缺点：

1. 干掉了浏览器的back按钮，当然可以使用H5 history API。
2. Ajax因为暴露了更多的服务器逻辑，所以存在一定的安全风险。
3. 不利用SEO。
4. 破坏了url和资源统一的初衷。

请求预编码

function encodeFormData(data){

if(!data){

return "";

}

var pairs=[];

for(var key in data){

if(!data.hasOwnProperty(key)){

continue;

}

else if(typeof data[key] ==='function'){

continue;

}else {

var name=encodeURIComponent(key);

var value=encodeURIComponent(data[key].toString());

pairs.push(name+'='+value);

}

}

return pairs.join('&');

}

var xhr=new XMLHttpRequest();

xhr.open('GET',"http://www.baidu.com",true);

xhr.setRequestHeader("Content-Type","text/plain");

xhr.onreadystatechange=function(e){

if(xhr.readyState===4&&xhr.status===200){

var type=xhr.getResponseHeader("Content-Type");

if(type.match(/^text/)){

xhr.responseText;

}

}

}

xhr.send();//GET方式下，send没有参数或者传递null

//0 open未打开

//1 open已打开

//2 接收到请求头

//3 接收到请求体

//4 请求完成

新规范中，XHR对象在不同的阶段会触发不同的事件，所以不再需要检查readyState的值。

1. 当send（）发出消息之后，触发loadstart事件。
2. 当接受服务器端的消息的时候，会触发progress事件，如果请求速度过快，也有可能不触发progress事件。
3. 当请求完成的时候，触发load事件。
4. 请求没有完成的时候，会触发timeout、abort（中止）、error事件中的一个。
5. xhr.onprogress=function(e){
6. if(e.lengthComputable){
7. var x=e.loaded\*100/e.total;
8. }
9. }
10. //如果传输的内容长度可知，Content-Length可知，如果不可知则为0；
11. //事件对象event.lengthComputable为true
12. //event.loaded为已经下载的
13. //event.total为全部的

上传进度事件？

很奇怪不是直接在XHR对象上监控事件，而是XHR对象上有一个upload属性，它指向一个对象。这个对象的progress事件指的就是上传进度事件。

//上传事件

xhr.upload.onprogress=function(e){

if(e.lengthComputable){

var x=e.loaded\*100/e.total;

}

}

//上传完成事件

xhr.upload.onload=function(e){

}

CORS的时候，当你在xhr.open（）的第四个和第五个参数传入用户名和密码，绝对不会通过，因为这种方式让分布式密码攻击成为可能。其次，跨域的时候是不会传输cookie的，当然你可以设置xhr.withCredentials为true。

封装一个JSONP请求？

function getJSONP(url,callback){

var cbnum='cb'+getJSONP.count++;

var callbackName='getJSONP.'+cbnum;

if(url.indexOf("?")===-1){

url+='?callback='+callbackName;

}else{

url+='&callback='+callbackName;

}

var script= document.createElement('script');

getJSONP[callbackName]=function(response){

try{

callback(response);

}

finally{

delete getJSONP[callbackName];

script.parentNode.removeChild(script);

}

}

script.src=url;

document.body.appendChild(script);

}

getJSONP.count=0;

Window对象的hashchange事件和