AJAX原理（四部曲）

为什么使用Ajax?
以登录为例，普通的表单提交时会进行页面跳转，而且数据容易丢失，ajax不会，用户体验号
Ajax应用程序的优势在于：
1. 通过异步模式，提升了用户体验
2. 优化了浏览器和服务器之间的传输，减少不必要的数据往返，减少了带宽占用
3. Ajax引擎在客户端运行，承担了一部分本来由服务器承担的工作，从而减少了大用户量下的服务器负载。
2、AJAX最大的特点是什么。
Ajax可以实现动态不刷新（局部刷新）
就是能在不更新整个页面的前提下维护数据。这使得Web应用程序更为迅捷地回应用户动作，并避免了在网络上发送那些没有改变过的信息

```
//1、创建对象
let xhr = new XMLHttpRequest();
newActiveXObject("Microsoft.XMLHTTP" )IE6

//2、连接（请求）
xhr.open('GET','url',true); //true异步，false同步
```

get方式

xhr.open("get","login.php?aa=1&bb=2&cc=3",true/false)配置 第三参数 是否异步

get请求 会把要提交的信息（aa=1&bb=2&cc=3"）添加到地址栏的后面速度快，配置简 单不安全

post方式

```
xhr.open("post","login.php",true/false)配置
xhr.setRequestHeader("Content-Type","application/x-www-form-urlencoded");//将我们的请求添加到头部报文中
```

post请求 安全性比较高 配置复杂一些 传递一些数据量比较大的数据

```
//3、发送
xhr.send('body数据');发送请求
xhr.send("aa=1&bb=2&cc=3")

//4、接收
xhr.onreadystatechange = function(){
//onreadystatechange分多次执行
//readyState当前通信状态值：
//1、 0 初始状态：xhr对象刚刚创建完
//2、 1 连接：连接到服务器
//3、 2 发送请求：刚刚Send完
//4、 3 接收完成：头接收完了
//5、 4 接收完成：体接收完了
//status--http状态码，表明通信结果
//1xx    消息
//200，成功
//3xx  重定向
301 Moved Permanently(永久重定向，下回不会再请求这个服务器)
302-临时重定向（下回依然会请求这个服务器）
303-告知客户端使用另一个 URL来获取资源
 304-Not Modified（date 缓存未过期、缓存过期）
//4xx  请求错误，主要错误原因在客户端
400-请求格式错误。
1)语义有误，当前请求无法被服务器理解。除非进行修改，否则客户端不应该重复提交这个请求；
2)请求参数有误
//5xx  服务端错误
//6xx+ 自定义

if(xhr.readyState==4){
                if(xhr.status>=200&&xhr.status<300||xhr.status==304){
```

```
                        //success
                        console.log(xhr.response)
                        //xhr.response
                        //xhr.responseText 以文本方式返回数据
                        //xhr.responseURL
                        //xhr.responseXML  以XML方式返回数据
                        let arr = eval('('+xhr.responseText+')') //解析
方式1,不安全
                        let json=null;
                        try{
                                json = JSON.parse(xhr.respinseText) //
解析成json
                        }catch(e){
                                json = eval('('+xhr.responseText+')')
                        }
                }else {
                        //failed
                }
}
}
```

1.json.parse()/json.stringify()/eval()

eval() 函数可计算某个字符串，并执行其中的的 JavaScript 代码

2.responseText()

3.responseXml()

平时都接触过哪些ajax提交方式，一般什么时候用什么解决方式？

- get 一般用于获取数据的时候
- post 用于创建数据的时候
- put 用于修改数据（不严格的时候也可以不用）
- delete 用于删除数据

ajax中的get和post区别？

1. Get 方法通过 URL 请求来传递用户的数据，将表单内各字段名称与其内容，以成对的字符串连接；Post 方法通过 HTTP post 机制，将表单内各字段名称与其内容放置在 HTML 表头(header)内一起传送给服务器端

2. Get 方式传输的数据量非常小，一般限制在 2 KB 左右，但是执行效率却比 Post 方法好；而 Post 方式传递的数据量相对较大

3. 安全问题：get没有post安全

简述 AJAX的交互模型，以及同步和异步的区别

同步：脚本会停留并等待服务器发送回复然后继续。

异步：脚本不停留并处理可能的回复。

axios是什么？

请求后台资源的模块

和ajax有相同的用法，用于数据的交互

Axios.post().then()

Axios.get().then()

Ajax/axios/fetch的区别

1.jQuery ajax

```
$.ajax({
    type: 'POST',
    url: url,
    data: data,
    dataType: dataType,
    success: function () {},
    error: function () {}
});
```

优缺点：

1.本身是针对MVC的编程,不符合现在前端MVVM的浪潮

2.基于原生的XHR(XMLHttpRequest)开发，XHR本身的架构不清晰，个请求之间如果有先后关系的话，就会出现回调地狱,已经有了fetch的替代方案

3.JQuery整个项目太大，单纯使用ajax却要引入整个JQuery非常的不合理（采取个性化打包的方案又不能享受CDN服务

4. ajax 是对原生XHR的封装，除此以外还增添了对JSONP的支持,非常方便

5.不符合关注分离（Separation of Concerns）的原则

6.配置和调用方式非常混乱，而且基于事件的异步模型不友好。

2.axios

```
axios({
        method: 'post',
        url: '/user/12345',
        data: {
                firstName: 'Fred',
                lastName: 'Flintstone'
        }
})
.then(function (response) {
        console.log(response);
})
.catch(function (error) {
        console.log(error);
});
```

axios 是一个基于Promise 用于浏览器和 nodejs 的 HTTP 客户端，本质上也是对原生XHR的封装，只不过它是Promise的实现版本，符合最新的ES规范，它本身具有以下特征：

1.从浏览器中创建 XMLHttpRequest

2.支持 Promise API

3.客户端支持防止CSRF

4.提供了一些并发请求的接口（重要，方便了很多的操作）

5.从 node.js 创建 http 请求

6.拦截请求和响应

7.转换请求和响应数据

8.取消请求

9.自动转换JSON数据

PS:防止CSRF:就是让你的每个请求都带一个从cookie中拿到的key，根据浏览器同源策略，假冒的网站是拿不到你cookie中得key的，这样，后台就可以轻松辨别出这个请求是否是用户在假冒网站上的误导输入，从而采取正确的策略。

3.fetch

```
try {
    let response = await fetch(url);
    let data = response.json();
    console.log(data);
} catch(e) {
    console.log("Oops, error", e);
}
```

是在ES6出现的，使用了ES6中的promise对象，Fetch是基于promise设计的。

Fetch的代码结构比起ajax简单多了，参数有点像jQuery ajax。但是，一定记住fetch不是ajax的进一步封装，而是原生js，没有使用XMLHttpRequest对象。

fetch的优点：

1.符合关注分离，没有将输入、输出和用事件来跟踪的状态混杂在一个对象里

2.更好更方便的写法

我认为fetch的优势主要优势就是：

1. 语法简洁，更加语义化
2. 基于标准 Promise 实现，支持 async/await
3. 同构方便，使用

4.更加底层，提供的API丰富（request, response）

5.脱离了XHR，是ES规范里新的实现方式

最近在使用fetch的时候，也遇到了不少的问题：

fetch是一个低层次的API，你可以把它考虑成原生的XHR，所以使用起来并不是那么舒服，需要进行封装。

例如：

1）fetch只对网络请求报错，对400，500都当做成功的请求，服务器返回 400，500 错误码时并不会 reject，只有网络错误这些导致请求不能完成时，fetch 才会被 reject。

2）fetch默认不会带cookie，需要添加配置项： fetch(url, {credentials(证书）: 'include'})

3）fetch不支持abort，不支持超时控制，使用setTimeout及Promise.reject的实现的超时控制并不能阻止请求过程继续在后台运行，造成了流量的浪费

4）fetch没有办法原生监测请求的进度，而XHR可以

总结：axios既提供了并发的封装，也没有fetch的各种问题，而且体积也较小，当之无愧现在最应该选用的请求的方式。