# Jquery ajax, Axios, Fetch区别之我见

## 引言

前端技术真是一个发展飞快的领域，我三年前入职的时候只有原生XHR和Jquery ajax，我们还曾被JQuery 1.9版本版本以下不支持大文件请求这个问题卡了半天（最后自己写了原生的XHR请求）。一晃眼，JQuery ajax早已不能专美于前，axios和fetch都已经开始分别抢占“请求”这个前端高地。本文将会尝试着阐述他们之间的区别，并给出自己的一些理解。

## 1 JQuery ajax

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

这个我就不用多言了把，是对原生XHR的封装，除此以外还增添了对JSONP的支持。有一说一的说一句，JQuery ajax经过多年的更新维护，真的已经是非常的方便了，优点无需多言；如果是硬要举出几个缺点，那可能只有

- 本身是针对MVC的编程,不符合现在前端MVVM的浪潮
- 基于原生的XHR开发，XHR本身的架构不清晰，已经有了fetch的替代方案
- 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）

尽管JQuery对我们前端的开发工作曾有着（现在也仍然有着）深远的影响，但是我们可以看到随着VUE，REACT新一代框架的兴起，以及ES规范的完善，更多API的更新，JQuery这种大而全的JS库，未来的路会越走越窄。

> 总结：廉颇老矣，尚能饭，但是总有饭不动的一天。

# 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);
});
```

Vue2.0之后，尤雨溪推荐大家用axios替换JQuery ajax，想必让Axios进入了很多人的目光中。Axios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，从它的官网上可以看到它有以下几条特性：

- 从 node.js 创建 http 请求
- 支持 Promise API
- 客户端支持防止CSRF
- **提供了一些并发请求的接口（重要，方便了很多的操作）**

这个支持防止CSRF其实挺好玩的，是怎么做到的呢，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。

Axios既提供了并发的封装，也没有下文会提到的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。

> 总结：谁敢横刀立马，唯我Axios将军！

# 3 Fetch

fetch号称是AJAX的替代品，它的好处在《传统 Ajax 已死，Fetch 永生》中

提到有以下几点：

- 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
- 更好更方便的写法，诸如：

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

坦白说，上面的理由对我来说完全没有什么说服力，因为不管是Jquery还是Axios都已经帮我们把xhr封装的足够好，使用起来也足够方便，为什么我们还要花费大力气去学习fetch？

我认为fetch的优势主要优势就是：

- 更加底层，提供的API丰富（request, response）
- 脱离了XHR，是ES规范里新的实现方式

大家都喜欢新的东西，坦白说，作为一个前端工程师，我在使用原生XHR的时候，尽管偶尔觉得写的丑陋，但是在使用了JQuery和axios之后，已经对这一块完全无所谓了。当然，如果新的fetch能做的同样好，我为了不掉队也会选择使用fetch。这个道理其实很好理解：你有一架歼8，魔改了N次，性能达到了歼10的水准，但是要是有个人给你拿来一架新的歼10，你也会毫不犹豫的选择新的歼10——不仅仅是新，也代表了还有新的魔改潜力。

但是我最近在使用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，需要封装去处理

> 2）fetch默认不会带cookie，需要添加配置项
> 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

PS: fetch的具体问题大家可以参考：《fetch没有你想象的那么美》《fetch使用的常见问题及解决方法》

看到这里，你心里一定有个疑问，这鬼东西就是个半拉子工程嘛，我还是回去用Jquery或者Axios算了——其实我就是这么打算的。但是，必须要提出的是，我发现fetch在前端的应用上有一项xhr怎么也比不上的能力：跨域的处理。

我们都知道因为同源策略的问题，浏览器的请求是可能随便跨域的——一定要有跨域头或者借助JSONP，但是，fetch中可以设置mode为"no-cors"（不跨域），如下所示：

```
fetch('/users.json', {
    method: 'post',
    mode: 'no-cors',
    data: {}
}).then(function() { /* handle response */ });
```

这样之后我们会得到一个type为“opaque”的返回。需要指出的是，这个请求是真正抵达过后台的，所以我们可以使用这种方法来进行信息上报，在我们之前的image.src方法中多出了一种选择，另外，我们在network中可以看到这个请求后台设置跨域头之后的实际返回，有助于我们提前调试接口（当然，通过chrome插件我们也可以做的到）。总之，fetch现在还不是很好用，我尝试过几个fetch封装的包，都还不尽如人意。

> 总结：酋长的孩子，还需成长

# 总结

如果你是直接拉到文章底部的，只需要知道现在无脑使用axios即可，Jquery老迈笨拙，fetch年轻稚嫩，只有Axios正当其年！