

AJAX 课程

尚硅谷前端研究院

版本：V 1.0

第 1 章：原生 AJAX

1.1 AJAX 简介

AJAX 全称为 Asynchronous JavaScript And XML，就是异步的 JS 和 XML。

通过 AJAX 可以在浏览器中向服务器发送异步请求，最大的优势：**无刷新获取数据。**

AJAX 不是新的编程语言，而是一种将现有的标准组合在一起使用的新方式。

1.2 XML 简介

XML 可扩展标记语言。

XML 被设计用来传输和存储数据。

XML 和 HTML 类似，不同的是 HTML 中都是预定义标签，而 XML 中没有预定义标签，全都是自定义标签，用来表示一些数据。

比如说我有一个学生数据：

```
name = "孙悟空"; age = 18; gender = "男";
```

用 XML 表示：

```
<student>
  <name>孙悟空</name>
  <age>18</age>
  <gender>男</gender>
</student>
```

现在已经被 JSON 取代了。

用 JSON 表示：

```
{"name":"孙悟空","age":18,"gender":"男"}
```

1.3 AJAX 的特点

1.3.1 AJAX 的优点

- 1) 可以无需刷新页面而与服务器端进行通信。
- 2) 允许你根据用户事件来更新部分页面内容。

1.3.2 AJAX 的缺点

- 1) 没有浏览历史，不能回退
- 2) 存在跨域问题(同源)
- 3) SEO 不友好

1.4 AJAX 的使用

1.4.1 核心对象

XMLHttpRequest，AJAX 的所有操作都是通过该对象进行的。

1.4.2 使用步骤

- 1) 创建 XMLHttpRequest 对象

```
var xhr = new XMLHttpRequest();
```

- 2) 设置请求信息

```
xhr.open(method, url);
```

```
//可以设置请求头，一般不设置
```

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

3) 发送请求

```
xhr.send(body) //get 请求不传 body 参数，只有 post 请求使用
```

4) 接收响应

```
//xhr.responseXML 接收 xml 格式的响应数据
```

```
//xhr.responseText 接收文本格式的响应数据
```

```
xhr.onreadystatechange = function (){  
    if(xhr.readyState == 4 && xhr.status == 200){  
        var text = xhr.responseText;  
        console.log(text);  
    }  
}
```

1.4.3 解决 IE 缓存问题

问题：在一些浏览器中(IE),由于缓存机制的存在，ajax 只会发送的第一次请求，剩余多次请求不会在发送给浏览器而是直接加载缓存中的数据。

解决方式：浏览器的缓存是根据 url 地址来记录的，所以我们只需要修改 url 地址即可避免缓存问题

```
xhr.open("get", "/testAJAX?t="+Date.now());
```

1.4.4 AJAX 请求状态

xhr.readyState 可以用来查看请求当前的状态

<https://developer.mozilla.org/zh-CN/docs/Web/API/XMLHttpRequest/readyState>

0: 表示 XMLHttpRequest 实例已经生成，但是 open()方法还没有被调用。

1: 表示 send()方法还没有被调用，仍然可以使用 setRequestHeader(), 设定 HTTP

请求的头信息。

- 2: 表示 `send()` 方法已经执行，并且头信息和状态码已经收到。
- 3: 表示正在接收服务器传来的 `body` 部分的数据。
- 4: 表示服务器数据已经完全接收，或者本次接收已经失败了

第 2 章：jQuery 中的 AJAX

2.1 get 请求

`$.get(url, [data], [callback], [type])`

url: 请求的 URL 地址。

data: 请求携带的参数。

callback: 载入成功时回调函数。

type: 设置返回内容格式，`xml`, `html`, `script`, `json`, `text`, `_default`。

2.2 post 请求

`$.post(url, [data], [callback], [type])`

url: 请求的 URL 地址。

data: 请求携带的参数。

callback: 载入成功时回调函数。

type: 设置返回内容格式，`xml`, `html`, `script`, `json`, `text`, `_default`。

第 3 章：跨域

3.1 同源策略

同源策略(Same-Origin Policy)最早由 Netscape 公司提出，是浏览器的一种安全策略。

同源： 协议、域名、端口号 必须完全相同。

违背同源策略就是跨域。

3.2 如何解决跨域

3.2.1 JSONP

1) JSONP 是什么

JSONP(JSON with Padding), 是一个非官方的跨域解决方案, 纯粹凭借程序员的聪明才智开发出来, 只支持 `get` 请求。

2) JSONP 怎么工作的?

在网页有一些标签天生具有跨域能力, 比如: `img link iframe script`。

JSONP 就是利用 `script` 标签的跨域能力来发送请求的。

3) JSONP 的使用

1. 动态的创建一个 `script` 标签

```
var script = document.createElement("script");
```

2. 设置 `script` 的 `src`, 设置回调函数

```
script.src = "http://localhost:3000/testAJAX?callback=abc";  
  
function abc(data) {  
    alert(data.name);  
};
```

3. 将 `script` 添加到 `body` 中

```
document.body.appendChild(script);
```

4. 服务器中路由的处理

```
router.get("/testAJAX", function (req, res) {  
    console.log("收到请求");  
    var callback = req.query.callback;  
    var obj = {
```

```
        name:"孙悟空",  
        age:18  
    }  
    res.send(callback+"("+JSON.stringify(obj)+")");  
});
```

4) jQuery 中的 JSONP

```
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <title>Title</title>  
</head>  
<body>  
    <button id="btn">按钮</button>  
    <ul id="list"></ul>  
    <script type="text/javascript" src="./jquery-1.12.3.js"></script>  
    <script type="text/javascript">  
        window.onload = function () {  
            var btn = document.getElementById('btn')  
            btn.onclick = function () {  
                $.getJSON("http://api.douban.com/v2/movie/in_theaters?callback=?",function  
(data) {  
                    console.log(data);  
                    //获取所有的电影的条目  
                    var subjects = data.subjects;  
                    //遍历电影条目
```

```
for(var i=0 ; i<subjects.length ; i++){  
    $("#list").append("<li>"+  
        subjects[i].title+"<br />"+  
        "<img src='"+subjects[i].images.large+"'>"+  
        "</li>");  
}  
});  
}  
}  
</script>  
</body>  
</html>
```

3.2.2 CORS

https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Access_control_CORS

1) CORS 是什么？

CORS (Cross-Origin Resource Sharing)，跨域资源共享。CORS 是官方的跨域解决方案，它的特点是不需要在客户端做任何特殊的操作，完全在服务器中进行处理，支持 get 和 post 请求。跨域资源共享标准新增了一组 HTTP 首部字段，允许服务器声明哪些源站通过浏览器有权访问哪些资源

2) CORS 怎么工作的？

CORS 是通过设置一个响应头来告诉浏览器，该请求允许跨域，浏览器收到该响应以后就会对响应放行。

3) CORS 的使用

主要是服务器端的设置：

```
router.get("/testAJAX" , function (req , res) {
```

```
//通过 res 来设置响应头，来允许跨域请求  
  
//res.set("Access-Control-Allow-Origin","http://127.0.0.1:3000");  
  
res.set("Access-Control-Allow-Origin","*");  
  
res.send("testAJAX 返回的响应");  
  
});
```