跨域几种方式

脚本之家 2018-11-06

作者 | 浪里行舟

来源 | segmentfault.com/a/1190000016756432

一、什么是跨域

JavaScript出于安全方面的考虑，不允许跨域调用其他页面的对象。那什么是跨域呢，简单地理解就是因为JavaScript同源策略的限制， `a.com` 域名下的js无法操作 `b.com` 或是 `c.a.com` 域名下的对象。

当协议、子域名、主域名、端口号中任意一个不相同时，都算作不同域。不同域之间相互请求资源，就算作“跨域”。例如：http://www.abc.com/index.html 请求 http://www.efg.com/service.php。

有一点必须要注意：**跨域并不是请求发不出去，请求能发出去，服务端能收到请求并正常返回结果，只是结果被浏览器拦截了**。之所以会跨域，是因为受到了同源策略的限制，同源策略要求源相同才能正常进行通信，即协议、域名、端口号都完全一致。

大家可以参照下图，有助于深入理解跨域。

特别说明两点：

第一：如果是协议和端口造成的跨域问题“前台”是无能为力的。

第二：在跨域问题上，域仅仅是通过“URL的首部”来识别而不会根据域名对应的IP地址是否相同来判断。“URL的首部”可以理解为“协议，域名和端口必须匹配”。

二、什么是同源策略及其限制

同源策略限制从一个源加载的文档或脚本如何与来自另一个源的资源进行交互。这是一个用于隔离潜在恶意文件的关键的安全机制。**它的存在可以保护用户隐私信息，防止身份伪造等(读取**

Cookie)。

同源策略限制内容有：

- Cookie、LocalStorage、IndexedDB 等存储性内容
- DOM 节点
- AJAX 请求不能发送

但是有三个标签是允许跨域加载资源：

1. `<imgsrc=XXX>`
2. `<linkhref=XXX>`
3. `<scriptsrc=XXX>`

接下来我们讨论下有哪些处理跨域的方法。但所有的跨域都必须经过信息提供方的允许。如果未经允许即可获取，那是浏览器同源策略出现漏洞。

三、处理跨域方法一——JSONP

1.JSONP原理

利用 `<script>` 元素的这个开放策略，网页可以得到从其他来源动态产生的 JSON 数据。JSONP请求一定需要对方的服务器做支持才可以。

2.JSONP和AJAX对比

JSONP和AJAX相同，都是客户端向服务器端发送请求，从服务器端获取数据的方式。但AJAX属于同源策略，JSONP属于非同源策略（跨域请求）。

3.JSONP优缺点

JSONP优点是兼容性好，可用于解决主流浏览器的跨域数据访问的问题。**缺点是仅支持get方法具有局限性。**

4.JSONP的流程（以第三方API地址为例，不必考虑后台程序）

- 声明一个回调函数，其函数名(如fn)当做参数值，要传递给跨域请求数据的服务器，函数形参为要获取目标数据(服务器返回的data)。
- 创建一个 `<script>`标签，把那个跨域的API数据接口地址，赋值给script的src，还要在这个地址中向服务器传递该函数名（可以通过问号传参 `:?callback=fn`）。

    - 服务器接收到请求后，需要进行特殊的处理：把传递进来的函数名和它需要给你的数据拼接成一个字符串，例如：传递进去的函数名是fn，它准备好的数据是`fn([{"name":"jianshu"}])`。
- 最后服务器把准备的数据通过HTTP协议返回给客户端，客户端再调用执行之前声明的回调函数（fn），对返回的数据进行操作。

```
<script type="text/javascript">
    function fn(data) {
        alert(data.msg);
    }
</script>
<script type="text/javascript" src="http://crossdomain.com/jsonServerResponse?j
```

其中 fn 是客户端注册的回调的函数，目的获取跨域服务器上的json数据后，对数据进行在处理。

最后服务器返回给客户端数据的格式为：

```
fn({ msg:'this  is  json  data'})
```

5.jQuery的jsonp形式

JSONP都是GET和异步请求的，不存在其他的请求方式和同步请求，且jQuery默认就会给JSONP的请求清除缓存。

```
$.ajax({
url:"http://crossdomain.com/jsonServerResponse",
dataType:"jsonp",
type:"get",//可以省略
jsonpCallback:"fn",//->自定义传递给服务器的函数名，而不是使用jQuery自动生成的，可省略
jsonp:"jsonp",//->把传递函数名的那个形参callback变为jsonp，可省略
success:function (data){
console.log(data);}
});
```

四、处理跨域方法二——CORS

1.CORS原理

整个CORS通信过程，都是浏览器自动完成，不需要用户参与。对于开发者来说，CORS通信与同源的AJAX通信没有差别，代码完全一样。浏览器一旦发现AJAX请求跨源，就会自动添加一些附加的头信息，有时还会多出一次附加的请求，但用户不会有感觉。**因此，实现CORS通信的关键是服务器。只要服务器实现了CORS接口，就可以跨源通信。**

2.CORS优缺点

CORS要求浏览器(>IE10)和服务器的同时支持，是跨域的根本解决方法，由浏览器自动完成。

优点在于功能更加强大支持各种HTTP Method，缺点是兼容性不如JSONP。

只需要在服务器端做一些小小的改造即可：

```
header("Access-Control-Allow-Origin:*");
header("Access-Control-Allow-Methods:POST,GET");
```

例如：网站 `http://localhost:63342/` 页面要请求 `http://localhost:3000/users/userlist` 页面，userlist页面返回json字符串格 `{name:'Mr.Cao',gender:'male',career:'IT Education'}`：

```
//在服务器端设置同源策略地址
router.get("/userlist", function (req, res, next) {
    var user = {name: 'Mr.Cao', gender: 'male', career: 'IT Education'};
    res.writeHeader(200,{"Access-Control-Allow-Origin":'http://localhost:63342'
    res.write(JSON.stringify(user));
    res.end();
});
```

在响应头上添加 `Access-Control-Allow-Origin` 属性，指定同源策略的地址。同源策略默认地址是网页的本身。**只要浏览器检测到响应头带上了CORS，并且允许的源包括了本网站，那么就不会拦截请求响应**。

五、处理跨域方法三——WebSocket

Websocket是HTML5的一个持久化的协议，它实现了浏览器与服务器的全双工通信，同时也是跨域的一种解决方案。WebSocket和HTTP都是应用层协议，都基于 TCP 协议。但是 WebSocket 是一种双向通信协议，在建立连接之后，WebSocket 的 server 与 client 都能主动向对方发送或接收数据。同时，WebSocket 在建立连接时需要借助 HTTP 协议，连接建立好了之后 client 与 server 之间的双向通信就与 HTTP 无关了。

原生WebSocket API使用起来不太方便，我们使用Socket.io，它很好地封装了webSocket接口，提供了更简单、灵活的接口，也对不支持webSocket的浏览器提供了向下兼容。

```
//前端代码：
<div>user input：<input type="text"></div>
<script src="./socket.io.js"></script>
<script>
var socket = io('http://www.domain2.com:8080');
// 连接成功处理
socket.on('connect', function() {
    // 监听服务端消息
    socket.on('message', function(msg) {
        console.log('data from server: ---> ' + msg);
    });

 // 监听服务端关闭
    socket.on('disconnect', function() {
        console.log('Server socket has closed.');
    });
});
document.getElementsByTagName('input')[0].onblur = function() {
    socket.send(this.value);
};
</script>
```

```
//Nodejs socket后台：
var http = require('http');
var socket = require('socket.io');
// 启http服务
var server = http.createServer(function(req, res) {
    res.writeHead(200, {
        'Content-type': 'text/html'
    });
    res.end();
});
server.listen('8080');
console.log('Server is running at port 8080...');
// 监听socket连接
socket.listen(server).on('connection', function(client) {
    // 接收信息
    client.on('message', function(msg) {
        client.send('hello：' + msg);
        console.log('data from client: ---> ' + msg);
    });

    // 断开处理
    client.on('disconnect', function() {
        console.log('Client socket has closed.');
```

```
    });
});
```

六、处理跨域方法四——postMessage

如果两个网页不同源，就无法拿到对方的DOM。典型的例子是iframe窗口和 `window.open` 方法打开的窗口，它们与父窗口无法通信。HTML5为了解决这个问题，引入了一个全新的API：跨文档通信 API（Cross-document messaging）。这个API为window对象新增了一个 `window.postMessage` 方法，允许跨窗口通信，不论这两个窗口是否同源。

postMessage方法的第一个参数是具体的信息内容，第二个参数是接收消息的窗口的源（origin），即"协议 + 域名 + 端口"。也可以设为 `*`，表示不限制域名，向所有窗口发送。

接下来我们看个例子： `http://localhost:63342/index.html` 页面向 `http://localhost:3000/message.html` 传递“跨域请求信息”

```
//发送信息页面 http://localhost:63342/index.html
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>跨域请求</title>
</head>
<body>
    <iframe src="http://localhost:3000/users/reg" id="frm"></iframe>
    <input type="button" value="OK" onclick="run()">
</body>
</html>
<script>
   function  run(){
        var frm=document.getElementById("frm");
        frm.contentWindow.postMessage("跨域请求信息","http://localhost:3000");
   }
</script>
```

```
//接收信息页面 http://localhost:3000/message.html
 window.addEventListener("message",function(e){  //通过监听message事件，可以监听对方
  console.log(e.data);
},false);
```

七、参考文章

- 跨域资源共享 CORS 详解
- 浏览器同源政策及其规避方法

- 你了解跨域请求吗?
- 前端常见跨域解决方案（全）

今日头条回顾：

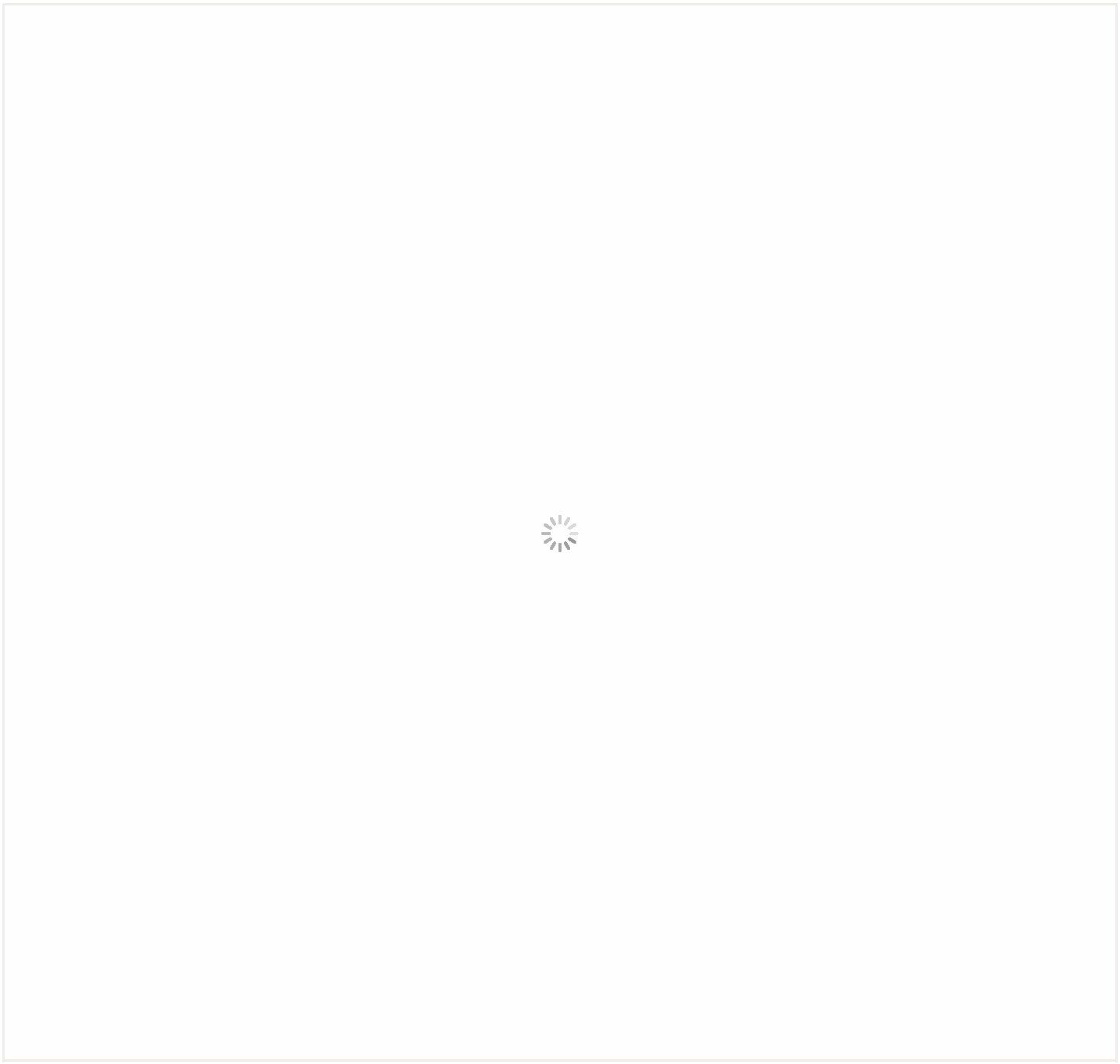
编程小工具，让你的编程之路如虎添翼！

记一次前端大厂面试

送给你优秀的技术学习资源

值得收藏的HTML5 Canvas动画特效集合

- 作为程序员，最起码要知道的几个公众号
- 分享7款超赞的CSS3动画效果，免费获取！
- 99%的程序员都会收藏的书单



小贴士

返回 上一级 搜索“Java 女程序员 大数据 留言送书 运维 算法 Chrome 黑客 Python JavaScript 人工智能 女朋友 MySQL 书籍 等关键词获取相关文章推荐。