**关注”千锋互联”微信公众号( ID: qianfengjiaoyu ),更多学习资源(电子书/面试题/简历模板/视频/公开课等等)与你分享**



**必须理解的跨域请求方式**

**什么是跨域？**

同源策略是由Netscape提出的著名安全策略，是浏览器最核心、基本的安全功能,它限制了一个源(origin)中加载文本或者脚本与来自其他源(origin)中资源的交互方式  
，所谓的同源就是指协议、域名、端口相同。  
当浏览器执行一个脚本时会检查是否同源，只有同源的脚本才会执行，如果不同源即为跨域

**跨域的几种方式**

在项目中可能会需要在一个域名下请求另外一个域名的资源，下面我们来探讨下跨域的几种实现方式

**1、jsonp**

最常见的一种跨域方式，其背后原理就是利用了script标签不受同源策略的限制，在页面中动态插入了script，script标签的src属性就是后端api接口的地址，并且以get的方式将前端回调处理函数名称告诉后端，后端在响应请求时会将回调返还，并且将数据以参数的形式传递回去。

前端:

//http://127.0.0.1:8888/jsonp.html

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

script.src = 'http://127.0.0.1:2333/jsonpHandler?callback=\_callback'

document.body.appendChild(script); //插入script标签

//回调处理函数 \_callback

var \_callback = function(obj){

for(key in obj){

console.log('key: ' + key +' value: ' + obj[key]);

}

}

后端:

//http://127.0.0.1:2333/jsonpHandler

app.get('/jsonpHandler', (req,res) => {

let callback = req.query.callback;

let obj = {

type : 'jsonp',

name : 'weapon-x'

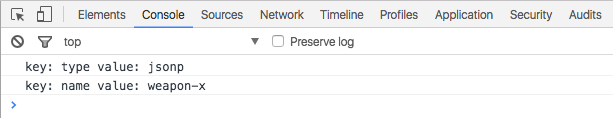
};

res.writeHead(200, {"Content-Type": "text/javascript"});

res.end(callback + '(' + JSON.stringify(obj) + ')');

})

在jsonp.html中打开控制台可以看到返回数据的输出:



jsonp.png

**2、CORS**

Cross-Origin Resource Sharing(跨域资源共享)是一种允许当前域（origin）的资源（比如html/js/web service）被其他域（origin）的脚本请求访问的机制。  
当使用XMLHttpRequest发送请求时，浏览器如果发现违反了同源策略就会自动加上一个请求头:origin,后端在接受到请求后确定响应后会在Response Headers中加入一个属性:Access-Control-Allow-Origin,值就是发起请求的源地址([http://127.0.0.1:8888)，浏览器得到响应会进行判断Access-Control-Allow-Origin的值是否和当前的地址相同，只有匹配成功后才进行响应处理。](http://127.0.0.1:8888)%EF%BC%8C%E6%B5%8F%E8%A7%88%E5%99%A8%E5%BE%97%E5%88%B0%E5%93%8D%E5%BA%94%E4%BC%9A%E8%BF%9B%E8%A1%8C%E5%88%A4%E6%96%ADAccess-Control-Allow-Origin%E7%9A%84%E5%80%BC%E6%98%AF%E5%90%A6%E5%92%8C%E5%BD%93%E5%89%8D%E7%9A%84%E5%9C%B0%E5%9D%80%E7%9B%B8%E5%90%8C%EF%BC%8C%E5%8F%AA%E6%9C%89%E5%8C%B9%E9%85%8D%E6%88%90%E5%8A%9F%E5%90%8E%E6%89%8D%E8%BF%9B%E8%A1%8C%E5%93%8D%E5%BA%94%E5%A4%84%E7%90%86%E3%80%82)

现代浏览器中和移动端都支持CORS（除了opera mini），IE下需要8+

前端:

//http://127.0.0.1:8888/cors.html

var xhr = new XMLHttpRequest();

xhr.onload = function(data){

var \_data = JSON.parse(data.target.responseText)

for(key in \_data){

console.log('key: ' + key +' value: ' + \_data[key]);

}

};

xhr.open('POST','http://127.0.0.1:2333/cors',true);

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

xhr.send();

后端:

//http://127.0.0.1:2333/cors

app.post('/cors',(req,res) => {

if(req.headers.origin){

res.writeHead(200,{

"Content-Type": "text/html; charset=UTF-8",

"Access-Control-Allow-Origin":'http://127.0.0.1:8888'

});

let people = {

type : 'cors',

name : 'weapon-x'

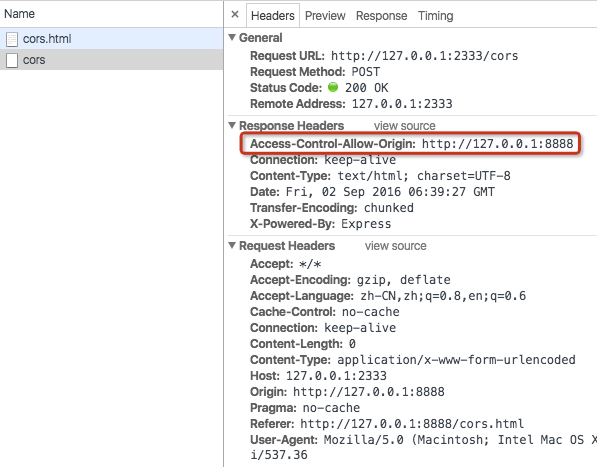
}

res.end(JSON.stringify(people));

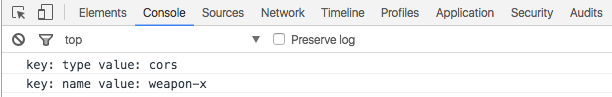
}

})

可以在开发者工具里面看到请求的详细信息，并且在控制台也可以看到返回的数据输出：



response header.png



cors console.png

**3、服务器跨域**

在前后端分离的项目中可以借助服务器实现跨域，具体做法是：前端向本地服务器发送请求，本地服务器代替前端再向api服务器接口发送请求进行服务器间通信，本地服务器其实就是个中转站的角色，再将响应的数据返回给前端，下面用node.js做一个示例

前端:

//http://127.0.0.1:8888/server

var xhr = new XMLHttpRequest();

xhr.onload = function(data){

var \_data = JSON.parse(data.target.responseText)

for(key in \_data){

console.log('key: ' + key +' value: ' + \_data[key]);

}

};

xhr.open('POST','http://127.0.0.1:8888/feXhr',true); //向本地服务器发送请求

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

xhr.send("url=http://127.0.0.1:2333/beXhr"); //以参数形式告知需要请求的后端接口

后端:

//http://127.0.0.1:8888/feXhr

app.post('/feXhr',(req,res) => {

let url = req.body.url;

superagent.get(url) //使用superagent想api接口发送请求

.end(function (err,docs) {

if(err){

console.log(err);

return

}

res.end(docs.res.text); //返回到前端

})

})

//http://127.0.0.1:2333/beXhr

app.get('/beXhr',(req,res) => {

let obj = {

type : 'superagent',

name : 'weapon-x'

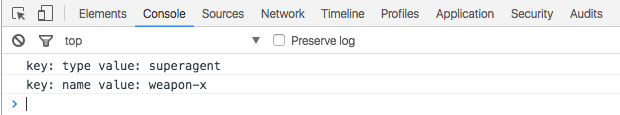
};

res.writeHead(200, {"Content-Type": "text/javascript"});

res.end(JSON.stringify(obj)); //响应

})

回到 <http://127.0.0.1:8888/server> 页面打开控制台可以看到数据输出：



console.png

**4、postmessage跨域**

在HTML5中新增了postMessage方法，postMessage可以实现跨文档消息传输（Cross Document Messaging），Internet Explorer 8, Firefox 3, Opera 9, Chrome 3和 Safari 4都支持postMessage。  
该方法可以通过绑定window的message事件来监听发送跨文档消息传输内容。  
使用postMessage实现跨域的话原理就类似于jsonp，动态插入iframe标签，再从iframe里面拿回数据  
，私认为用作跨页面通信更加适合

**总结**

当然还有其他实现跨域的方式比如在ie8、9下XDR跨域方案，flash方案，以上是一些常用的跨域方案，都各有利弊，比如：jsonp只能发送get请求、服务器跨域需要另起服务器等等，大家可以根据自己项目需求选择适合的解决方案.

**关注”千锋互联”微信公众号( ID: qianfengjiaoyu ),更多学习资源(电子书/面试题/简历模板/视频/公开课等等)与你分享**

