# Ajax入门和应用

第3天课堂笔记（本课程共5天）

# 目录

[Ajax入门和应用 1](#_Toc477332110)

[目录 1](#_Toc477332111)

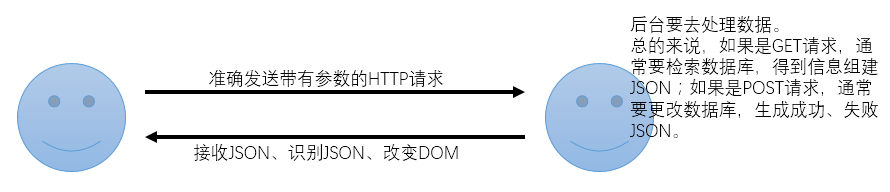
[一、复习 2](#_Toc477332112)

[二、JSONP跨域 4](#_Toc477332113)

[2.1 JSONP概念 4](#_Toc477332114)

[2.2 JSONP实用函数封装 7](#_Toc477332115)

# 一、复习



发送请求：

|  |
| --- |
| j.get("check.php",{"xuehao":199991},function(err,data){  }); |

|  |
| --- |
| j.post("check.php",{"xuehao":199991},function(err,data){  }); |

data返回的可能是JSON样子的字符串，所以我们要把这个字符串变为真正的对象JSON：

|  |
| --- |
| var jsonObj = eval("(" + data + ")");  var jsonObj = (new Function("return " + data))();  var jsonObj = JSON.parse(data); |

我们要会识别、解析JSON。枚举属性用.，枚举数组用[]。

|  |
| --- |
| {  "name" : "小明",  "sex" : "男",  "age" : 12,  "award" : [  {  "title" : "三好学生",  "time" : "2016年"  },  {  "title" : "三好学生",  "time" : "2015年"  },  {  "title" : "三好学生",  "time" : "2014年"  }  ]  } |

比如拿到2015年：

|  |
| --- |
| dataObj.award[1].time |

接下来就要通过JSON来对DOM进行处理，我们发现处理DOM（尤其是增加节点），不管是字符串方法、DOM方法都复杂，都需要连字符。而我们不喜欢连字符，提出模板技术。

|  |
| --- |
| // 模板字符串  var str = "我早上吃了<%=fan%>，花了<%=qian%>元，我很<%=xinqing%>啊！";  // 字典  var dictionary = {  "fan" : "煎饼",  "qian" : 4.5,  "xinqing" : "高兴"  }  // 数据绑定，（或者叫做数据填充，数据注入）  str = str.replace(/\<\%\=(\w+)\%\>/g,function(match,$1){  return dictionary[$1];  });  alert(str); |

所有的模板都写在

|  |
| --- |
| <script type="text/template" id="temp1"></script> |

里面，这是一个放模板的容器，可以通过

|  |
| --- |
| $("#temp1").html(); |

得到模板字符串。

前端Ajax技术，特别是模板技术，更改了企业里面的习惯。

2011年之前，后台处理模板

|  |
| --- |
| <?php  while($row = mysql\_fetch\_array($result)){  ?>  <div class="news">  <a href="**<?php echo $row["link"]; ?>**">  <div class="pic">  <img src="" alt="" />  </div>  <div class="content">  <h3>**<?php echo $row["title"]; ?>**</h3>  <div class="summary">**<?php echo $row["summary"]; ?>**</div>  <div class="info">  <span class="zuozhe">**<?php echo $row["link"]; ?>**</span>  <span class="shijian">**<?php echo $row["link"]; ?>**</span>  <span class="yuedu">阅读(**<?php echo $row["link"]; ?>**)</span>  </div>  </div>  </a>  </div>  <?php  }  ?> |

2011年之后，开始流行前后端分离。后台代码不要出现在前端！！后台只负责输入JSON，然后前端调用。

|  |
| --- |
| <script type="text/template" id="temp">  <div class="news">  <a href="<%=m\_display\_url%>">  <div class="pic">  <img src="<%=m\_image\_url%>" alt="" />  </div>  <div class="content">  <h3><%=m\_title%></h3>  <div class="summary"><%=m\_summary%></div>  <div class="info">  <span class="zuozhe"><%=m\_writer\_name%></span>  <span class="shijian"><%=m\_create\_time%></span>  <span class="yuedu"><%=hotcount%></span>  </div>  </div>  </a>  </div>  </script> |

革命式的里程碑：

table →→→→ div + css

后台模板 →→→→ 前端模板，前后端分离

前端开发工程师更重要的不是轮播图、浮动定位，在今天，必须要碰到数据！！

# 

# 二、JSONP跨域

## 2.1 JSONP概念

Ajax不能跨域，比如你的页面在127.0.0.1里，不能请求127.0.0.2里面的JSON文件。比如，你在www.163.com里，不能请求www.sohu.com里面的JSON文件。

域名不同（实际上ip不同也不行），Ajax不能跨域名的。比如news.sina.com.cn能不能请求ent.sina.com.cn不一定！如果news这个机器和ent是一个ip，就能，如果不是一个机器，就不能。

比如我们现在的页面运行在127.0.0.1，试图请求127.0.33.88的文件，会报错：

|  |
| --- |
| $.get("**http://127.0.33.88/**test.txt",function(data){  alert(data);  }); |



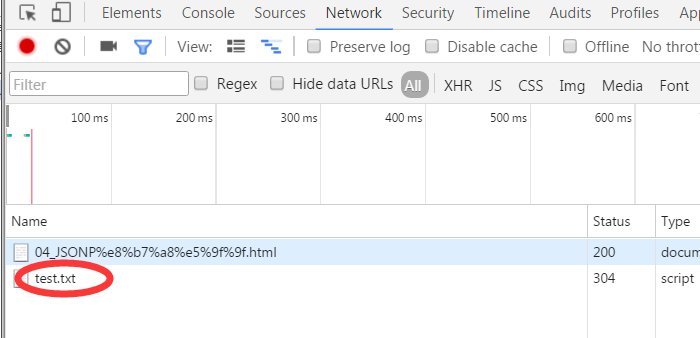
你觉得不能跨域这个事情，是浏览器的限制，还是服务器的限制？是服务器的限制，服务器有机制阻止来自其他网址的XHR请求。

比如A服务器上运行着页面，B服务器上有数据库（有JSON），现在A必须去请求B，怎么办？用JSONP。

JSONP不是Ajax！是一种奇淫技巧，发明人不可考，近些年变得流行。

首先，我们写一个script标签，这个标签现在有一个src属性，指向一个txt文件。此时将会发出HTTP请求，请求这个文件。

|  |
| --- |
| <script type="text/javascript" src="test.txt"></script> |



我们现在请求不变：

|  |
| --- |
| <script type="text/javascript" src="test.txt"></script> |

但是改变文件

|  |
| --- |
| alert("你好"); |

此时你好就能弹出，这是因为script标签会引包，同时也执行了js文件。

问题是，**HTML页面没有变量能够持有这个你好两个字**，所以我们现在就必须用奇淫技巧：

|  |
| --- |
| fun("你好"); |

我们之前都是在js文件中定义一个函数，在主页面中调用；而今天我们给反过来了！！在主页面定义函数，在js文件中调用函数！！

主页面中定义函数：

|  |
| --- |
| function fun(obj){  alert(obj.result[1].nianling);  } |

在外部文件中调用函数：

|  |
| --- |
| fun("你好"); |

此时外部文件的数据，被当做了实参传入了HTML页面，被形参接收。

如果参数是JSON呢？那JSON不就进页面了么！！我们起了一个名字，JSON with Padding。简称JSONP。

红色的部分（调用函数语句）就是所谓的padding。

|  |
| --- |
| **fun(**  {  "result" : [  {  "xingming":"小明",  "nianling" : 12,  "sex" : "男"  },  {  "xingming":"小红",  "nianling" : 13,  "sex" : "女"  },  {  "xingming":"小绿",  "nianling" : 9,  "sex" : "女"  }  ]  }  **);** |

JSONP是可以跨域的，垫付了Ajax的写法。甚至不用出现XMLHttpRequest对象就能实现Ajax。

JSONP刚刚诞生的时候，整个世界觉得被戏弄了。

JSONP是函数的调用，通过实参把数据传递到HTML页面中。

|  |
| --- |
| <script type="text/javascript">  function fun(obj){  **→ 定义函数**    }  </script>  <script type="text/javascript" src="test.txt"></script>  **→ 调用函数** |

JSONP不安全。因为如果你的JSONP文件，被人篡改了，你的页面就等于被黑了~~。

所以使用JSONP的时候，如果JSONP文件是自己公司的，放心大胆用，一点毛病没有！

## 2.2 JSONP实用函数封装

我们现在能不能封装一个类似：

|  |
| --- |
| myajax.get(URL,{},function(){  }); |

这样的函数。

主页面：

|  |
| --- |
| myjsonp("http://www.jd.com/sadfadsf/32rwe/wer",function(data){  //data就是JSONP请求回来的值  },"fetchJSON\_comment98vv21142"); |

JSONP外面的文件在调用函数，

|  |
| --- |
| fetchJSON\_comment98vv21142 ("你好"); |

这是我们已经写好的函数：

|  |
| --- |
| function myjsonp(URL,callback,callbackname){  //给系统中创建一个全局变量，叫做callbackname，指向callback函数  //定义  window[callbackname] = callback;  //创建一个script节点  var oscript = document.createElement("script");  //和image不一样，设置src并不会发出HTTP请求  oscript.src = URL;  oscript.type = "text/javascript";  //script标签的请求是在上树的时候发出，请求的是一个函数的执行语句  document.head.appendChild(oscript);  //为了不污染页面，我们瞬间把script拿掉  document.head.removeChild(oscript);  } |