# 服务器与Ajax

一、前端相关的技术点：

1、html（html5） 主要用来实现页面的排版布局

2、css（css3） 主要用来实现页面的样式美化

3、JavaScript（jQuery） 主要用来实现前端功能特效

采用上面的这些技术开发的页面和前端特效脚本需要放到服务器才能够对外提供服务，才能够让互联网上的网友看到。

二、初识Ajax

术语ajax最早产生于2005年，Ajax表示**Asynchronous JavaScript and XML(异步JavaScript和XML)**，但是它不是像HTML、JavaScript或CSS这样的一种“正式的”技术，它是表示一些技术的混合交互的一个术语（JavaScript、Web浏览器和Web服务器），它使我们可以获取和显示新的内容而不必载入一个新的Web页面。增强用户体验，更有桌面程序的感觉。

1、Ajax可以做什么？

显示新的HTML内容而不用载入整个页面、提交一个表单并且立即显示结果、登录而不用跳、转到新的页面、星级评定组件、遍历数据库信息加载更多而不刷新页面……

三、Ajax基础知识铺垫

1、客户端与服务器

它们之间的两者在逻辑上的组成是类似的。本质上都是计算机，只不过样子不同，配置不同，应用场景不同（安装的应用软件不同）

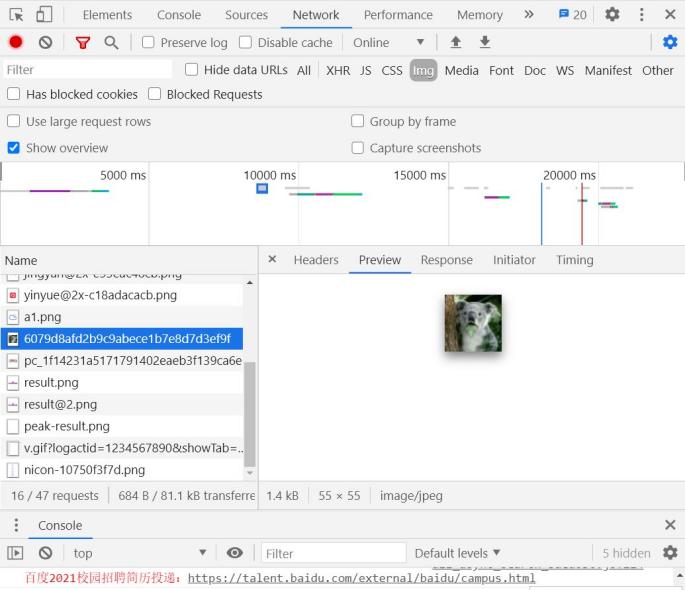
- 客户端主要用于普通上网用户。

- 服务器主要给上网用户提供后台服务。

（1）开发好的网页放在那里？

通过浏览器查看一个网页，使用调试工具查看页面信息

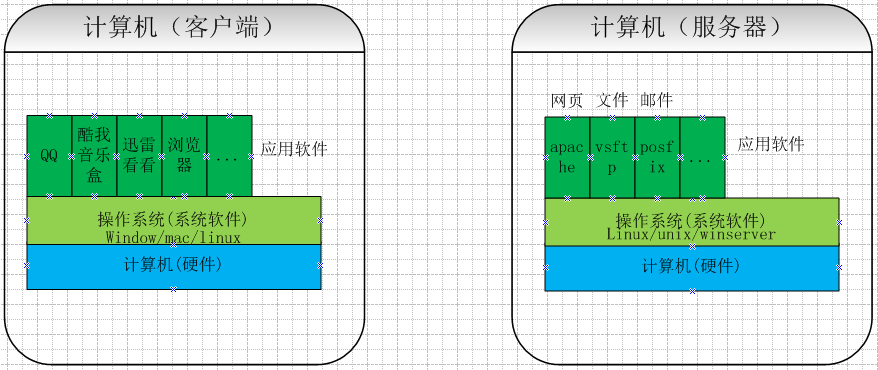
打开百度，F12网络，就可以看到网页请求的所有资源all，它里边的文件都是经过压缩处理，基本是看不懂什么意思



DOC就是页面中的所有标签，样式。

WS：webSocket在node中才会遇到。

（2）客户端与服务器对比



客户端的浏览器就会跟服务端提供的网页接口apache进行交互。

2、网络相关概念

IP地址、域名、端口、DNS

（1）IP地址：（唯一的确定互联网上的一台计算机）。

（2）域名：IP地址的别名，方便记忆。

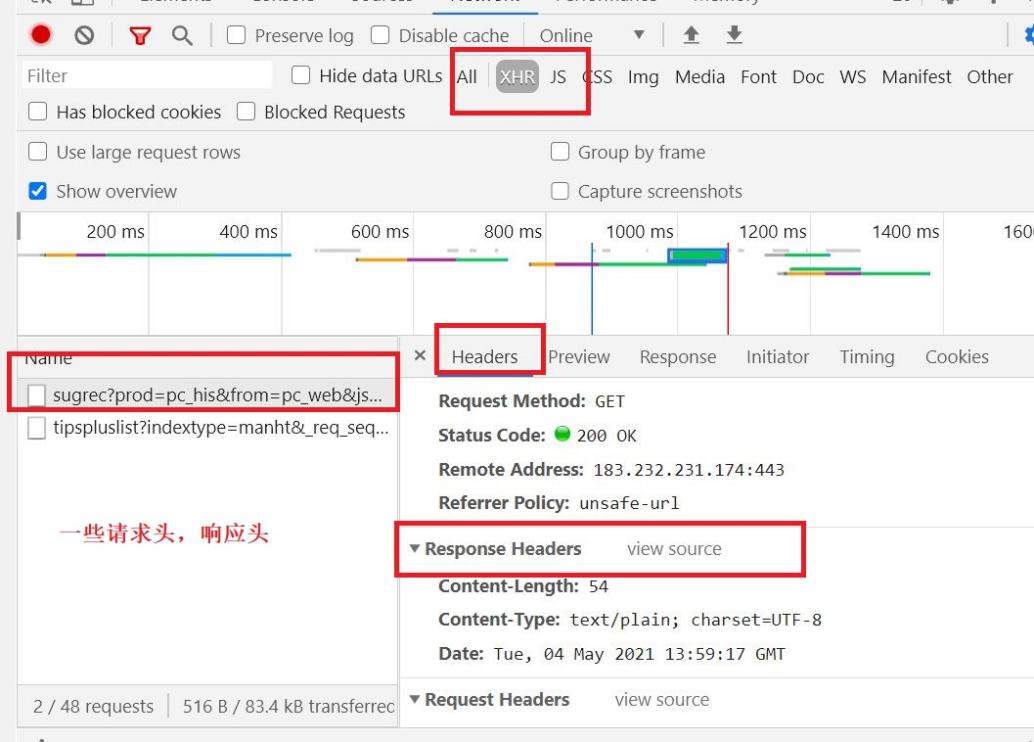
（3）DNS：用于维护IP地址与域名的关系。

（4）端口：用来确定计算机上的网络应用程序。

3、通信协议

http：超文本传输协议，网页HTML超文本标记语言，采用该协议。

我们只要记住http这种形式的url地址，可以向服务端发送一个请求，然后服务端给我们响应数据，我们拿到数据之后进行解析渲染，就可以实现一个特定的功能。这样差不多就够了。



- http/https 超为本传输协议

- ftp 文件传输协议

- smpt/pop3 邮件收发协议

4、服务器环境配置

这个集成环境主要用来提供网站服务。

localhost是本机的域名，默认的端口是80，http协议，IP是127.0.0.1，可以用cmd去ping localhost，就能看到地址。在谷歌中省略了http几个字符，复制url处理就能看到。

（1）wamp集成环境介绍

- windows：操作系统。

- Apache：提供静态资源服务（html页面、js文件、css文件、图片...）。

- MySQL：数据库。

- php：编程语言，可以用来开发网站。

（2）配置服务器网站的根目录路径

|  |
| --- |
| 1、找到文件D:\wamp\bin\apache\Apache2.4.4\conf\httpd.conf 或者打开如下文件（实际是同一个文件）  2   1. 在文件中搜索DocumentRoot   1  配置完成之后，一定要进行重启 |

（3）配置虚拟主机

配置虚拟主机可以配置多个网站（域名和网站目录对应），配置步骤如下

|  |
| --- |
| 1. 开启虚拟主机辅配置，在httpd.conf 中找到如下位置,然后把前面的井号去掉。   3   1. 配置虚拟主机，打开conf/extra/httpd-vhosts.conf   4  分别修改以下三项，其它项无需指定。  （1）DocumentRoot "E:/www/example"，这里的路径要在刚刚配置的网站的根目录下配置，否则无法找到。  （2）ServerName "example.com "、域名  （3）ServerAlias "www.example.com"、别名  4、修改DNS（hosts）文件(C:\Windows\System32\drivers\etc\hosts),  添加如下内容：  127.0.0.1 example.com  127.0.0.1 www.example.com  5、重启apache  访问http://www.example.com或者http://example.com  6、配置多个虚拟主机的实例如下：  httpd-vhosts.conf文件配置  5  hosts文件配置  6  在测试的时候，输入localhost，如果目录下有index.html那么它会默认显示这个  同个局域网内，是可以来访问wamp的，需要配置  （1）httpd.conf  搜索deny from All改成Allow，保存之后重启。 |

（4）手机端通过局域网访问

|  |
| --- |
| 1、首先查询到电脑的ip地址，在网络右键属性，可以找到。  1  2、查看到自己的ip，例如我是192.168.43.153   1. 添加虚拟主机，这里设置成ip地址，域名访问不到   <VirtualHost \*:80>  DocumentRoot "C:\Users\ASUS\Desktop\html\_study\workplace\Ajax\mobile"  ServerName 192.168.43.153  ServerAlias www.mobile.com  </VirtualHost>   1. system32中主机   127.0.0.1 192.168.43.153  5、重启，就ok |

四、网站

网站由一系列页面组成（页面由js、css、图片、html标签...所有的这些文件都被称为资源）。

1、静态网站

在url的最后是有后缀的html，这样的一般是静态网页。

就是提前写好的html页面（包括图片、媒体文件...静态资源文件），并且部署到服务器上。

（1）静态网站主要存在的问题：

随着网站规模的增大可维护性逐渐降低，改页面的东西，只能在服务器端去修改。

没有交互性，不能进行评价、登录等等。

2、动态网站

动态指的是html页面是动态生成的，这里动态生成的**不一定是一个完整的页面**，有可能仅仅是页面的一部分，或者仅仅是**数据**(普通字符串、json、xml)。

3、实现动态网站的技术，在页面当中嵌入一些标签。

- php

- java（jsp）

- .net

- Node.js

- python

- ......

4、两者的一些区别

有一些网站的url后边是一长串字符，并没有后缀，一般是动态网站，它们之间的区别在于这个页面是你提前写好的，还是动态生成的。



动态页面





动态页面是只有1个php的页面，他会对显示的数据进行一定的关连，例如点击橙子，就选择跟橙子相关图片、文字。如果页面要进行拓展的话，只要加一次代码就可以。有一套模板，然后用数据去填充

静态页面需要提前写好全部，他们都是比较相似的，只有图片和说明文字不同。如果需要在页面添加广告，那么所有的页面都要去写代码。

有个疑问，js不能处理这样的事情么？？？那假如页面不太一样，就再多加一个php的界面？

五、PHP弱类型语言

PHP（Pre Hypertext Preprocessor）即“超文本预处理器”，是在服务器端执行的脚本语言，尤其适用于Web开发并可嵌入HTML中。PHP语法利用了C、Java和Perl，该语言的主要目标是允许web开发人员快速编写动态网页。

|  |
| --- |
| PHP脚本是一种服务器端脚本程序，可通过嵌入等方法与HTML文件混合，也可以类，函数封装等形式，以模板的方式对用户请求进行处理。无论以何种方式，它的基本原理是这样的。  由客户端提出请求，**请求某一页面**　----->　WEB服务器引入指定相应脚本进行处理　----->　脚本被载入服务器　----->　由服务器指定的PHP解析器对脚本进行解析形成**HTML**语言形式　---->　将解析后的**HTML语句**以包的方式传回给浏览器。  由此不难看出，在页面发送到浏览器后，PHP就不存在了，已被转化解析为HTML语句。客户请求为一动态文件，事实上并没有真正的文件存在在那里，是PHP解析而成相对应的页面，然后发送回浏览器。这种页面处理方式被称为**“动态页面”**。  echo可以理解成html中的 write 方法。 |

php的数据类型与JavaScript的数据类型是类似的，都是弱类型语言，是由 PHP 根据该变量使用的上下文在运行时决定的。

在nework中查看服务器返回的数据，选all，服务器返回的是最终的一个html页面，没有php代码，展示出来的时候相当于是一个静态页面了，它生成的过程实际上是在服务器端进行的。具体的流程后续理解。在本机上是运行在apache环境中。

1、php基础语法

（1）所有的PHP代码都要放到该标签里**<?php ?>**

echo的作用就是向页面当中输入字符串。

|  |
| --- |
| 在服务器端的index.php代码  <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>PHP 页面</title>  </head>  <body>  <div>hello world</div>  <?php  echo '<div>hello i am PHP</div>';  ?>  </body>  </html>  1 |
| 1  不放在服务器端，浏览器是无法解析php文件的，会直接显示所有的代码字符。 |

2、变量

（1）$是变量的声明。

变量以$开头，然后接字母、数字、下划线，不能以数字开头，大小写敏感。

如：$num

（2）php中字符串的拼接用的是**.**

|  |
| --- |
| <div>hello world</div>  <?php  echo '<div>hello i am PHP</div>';  $num = 123;  // echo 'str'+ $num;无效  echo 'str'**.**$num; //有效  ?> |

（3）php中单引号对于其中的变量当作是普通的字符串来处理。

|  |
| --- |
| <div>hello world</div>  <?php  echo '<div>hello i am PHP</div>';  $num = 123;  // 无法解析变量值  echo 'str.$num.str';  ?> |

1. 双引号对于其中的变量会把变量解析成变量值。

|  |
| --- |
| <div>hello world</div>  <?php  echo '<div>hello i am PHP</div>';  $num = 123;  // 单引号无法解析变量值  echo "<div>str.$num.str</div>";  ?>  2 |

**注：php双引号不能解析二维数组。**

**注：双引号在解析键的时候，不需要加单引号或双引号。**

|  |
| --- |
| "<ul><li>姓名：$stuIdData[**name**]</li><li>语文：$stuIdData[**chinese**]</li><li>数学：$stuIdData[**math**]</li><li>英语：$stuIdData[**english**]</li></ul>" |

1. 内容输出

|  |
| --- |
| 1、 echo：输出简单数据类型，如字符串、数值  echo '\n';是无法识别转义字符的。  2、print\_r()：输出复杂数据类型，如数组    3、var\_dump()：输出详细信息，如对象、数组 |

3、php数组

（1）数组中如果单单传入一个值，那么它的键就是数字。

1. 传入键，那么在$arr[‘键名字’]，才能访问到，而不是数字。
2. 单值，又有键，那么无键的，它默认是键就是连续的数字。

数组的两种形式

$arr = array(1,2,3);默认key是从0开始的

$arr = array(‘a’=> ‘1’);

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>PHP 页面</title>  </head>  <body>  <div>  <?php  $arr = **array(1,22,32,43,5);**  echo $arr[0];  echo '<br />';  echo $arr[1];  echo '<br />';  echo '使用print\_r打印数组';  echo '<br />';  print\_r($arr);  1  $arr1 = **array('name'=>'lxf','age'=>'25','sex'=>'male');**  echo '<br />';  echo $arr1['name'];  echo '<br />';  echo '使用var\_dump打印数组';  echo '<br />';  var\_dump($arr1);  2  $arr2 = **array(1,2,3,'age'=>18,'hello');**  print\_r($arr2);  echo "<br />";  var\_dump($arr2);  ?>  3  </div>  </body>  </html> |

4、二维数组

数组中的每一项元素还是一个数组

|  |
| --- |
| <div>  <?php  $arr = array();  $arr[0] = array(1,2,3);  $arr[1] = array(4,5,6);  $arr[2] = array(7,8,9);  print\_r($arr);  /\*Array (  [0] => Array ( [0] => 1 [1] => 2 [2] => 3 )  [1] => Array ( [0] => 4 [1] => 5 [2] => 6 )  [2] => Array ( [0] => 7 [1] => 8 [2] => 9 ) )\*/  $arr1 = array(1,33);  $arr1['apple'] = array('color'=>'red','price'=>15);  $arr1['orange'] = array('color'=>'orange','price'=>35);  print\_r($arr1);  /\*Array (  [apple] => Array ( [color] => red [price] => 15 )  [orange] => Array ( [color] => orange [price] => 35 ) )\*/  /\*Array (  [0] => 1  [1] => 33  [apple] => Array ( [color] => red [price] => 15 )  [orange] => Array ( [color] => orange [price] => 35 ) )\*/  ?>  </div> |

5、数据类型

php的数据类型与JavaScript的数据类型是类似的，都是弱类型语言。

如果想查看某个[表达式](http://www.php.net/manual/zh/language.expressions.php)的值和类型，用 [var\_dump()](http://www.php.net/manual/zh/function.var-dump.php)。

如果只是想得到一个易读懂的类型的表达方式用于调试，用 [gettype()](http://www.php.net/manual/zh/function.gettype.php)。

要查看某个类型，不要用 [gettype()](http://www.php.net/manual/zh/function.gettype.php)，而用 is\_type 函数。

实际上 double 和 float 是相同的，由于一些历史的原因，这两个名称同时存在。

（1）字符串

（2）整型

（3）浮点型float （浮点型, 也称作 double)

（4）布尔型

（5）数组

（6）对象

（7）NULL

|  |
| --- |
| <div>  <?php  $a\_bool = true; // a boolean  $a\_str = "foo"; // a string  $a\_str2 = 'foo'; // a string  $an\_int = 12; // an integer  $a\_double = 1.1;  $a\_null = null;  echo (gettype($a\_null));  ?>  </div> |

6、函数

掌握一些基本概念就够前端使用了。

（1）自定义函数，函数名字大小写不敏感

|  |
| --- |
| 1、php函数也有预解析，函数可以先调用后声明。  <?php  $temp = saySomething('hello');  echo $temp;  function saySomething(**$info**) {  echo **$info**.'<br />';  return **$info**;  }  ?> |

（2）系统函数

gettype() 内置函数，用来判断变量的类型。

count() 是内置函数，用来计算数组的长度。

|  |
| --- |
| $a\_arr = array(1,2,3);  echo (gettype($a\_arr));  echo '<br />';  for ($i = 0; $i < count($a\_arr); $i++) {  echo $a\_arr[$i].'<br />';  }  输出1,2,3 |

foreach()

|  |
| --- |
| PHP foreach 循环结构是遍历数组时常用的方法，foreach 仅能够应用于数组和对象，如果尝试应用于其他数据类型的变量或者未初始化的变量将发出错误信息。  foreach 有以下两种语法格式：  //格式1 foreach (array\_expression as $value){     statement }  //格式2 foreach (array\_expression as $key => $value){     statement }  第一种格式遍历 array\_expression 数组时，每次循环将数组的值赋给 $value；  第二种遍历不仅将数组值赋给 $value，还将键名赋给 $key。 |
| $arr = array('name'=>'Mike','age'=>25,3);  foreach ($arr as $value) {  echo "$value <br/>";  }  foreach ($arr as $key => $value) {  echo "$key:$value <br />";  }  1 |
| 注：foreach 在 PHP5 和 PHP7 中的区别  在 PHP 5 版本中，当 foreach 开始循环执行时，每次数组内部指针都会自动向后移动一个单元，但是在 PHP 7 中却不是这样。如下代码在 PHP 5 和 PHP 7 中的执行结果会有所不同。  目前采用的是5版本，具体有遇到在研究。 |

json\_encode()

这个函数的功能是将数值转换成json形式的字符串，数据存储格式。

|  |
| --- |
| <?php  $arr = array(1,2,3,4);  echo json\_encode($arr);  echo "<br />";  $arr1 = array(**'name'=>'Mike','age'=>'14','sex'=>'male'**);  echo json\_encode($arr1);  ?> |

7、运算符

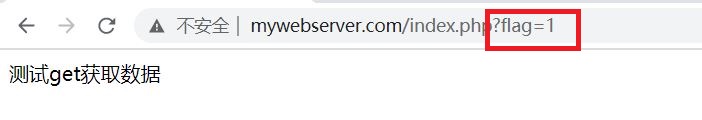
与JavaScript基本类似

8、预定义变量（表单处理）

可以理解成它跟内置的属性和内置的对象一样能直接使用。

他与前端与后端的交互有关系，比如你给后端传参数，以什么方式传过去，是以url参数传过去呢？还是以表单的方式传过去。前后端就是通过这两变量来进行交互的。

在输入url时候最后 ?flag=1，这样就把flag传到了后台



1. http协议常用的请求方式：（一般是用来对数据进行增删改查，不区分大小写）

get 用来从服务器获取数据（参数一般作为查询条件）

post 用来添加数据

put 用来修改数据

delete 用来删除数据

（2）$\_GET[ ]

$\_GET[‘xxx’]得到url地址中传递的参数xxx的值

|  |
| --- |
| <?php  $data = $\_GET['flag'];  echo '得到的url中的flag值：'.$data.'<br />';  var\_dump($data);  if ($data === '1') {  echo 'ok';  }  ?>  1  http://mywebserver.com/index.php?flag=1  默认是get |

（3）$\_POST[ ]

通过表单方式，表单传递数据用的就是name的属性，而不是id，通过submit提交到后台的一个页面上。

action就是把数据提交到哪个页面，它提交到的那个页面是后台的一个页面。

form默认的请求方式method就是get请求，get请求会把表单的数据作为url的参数。

|  |
| --- |
| html文件  <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>post测试</title>  </head>  <body>  <form **action**="http://www.myWebServer.com/index.php" **method**="post">  账号：<input type="text" name="username"><br />  密码：<input type="text" name="password"><br />  <input type="submit" value="登陆" id="btn">  </form>  </body>  </html> |
| 服务器php文件  <?php  $user = $\_POST['username'];  $pwd = $\_POST['password'];  // 告诉浏览器我返回的是文本类型，编码是utf-8的，设置服务器响应的文件类型，可以百度查mime，它返回是js，就使用对应js格式。  // 这里的text/plain要根据实际用途来设置，如果有html，就不能用这个，否则无法解析  header('content-Type**:**text**/**plain; charset**=**utf-8');  if ($user === 'admin' && $pwd === '123') {  echo 'ok成功';  } else {  echo 'no失败';  }  ?> |
|  |

php中的编码是utf-8，而浏览器它解码是采用GBK编码

url中没有传递的数据，这个数据实际上是在form data，这个叫做请求体，它是http协议的一部分，用get也可以，但是密码会暴露在url中，url可接受的长度大概是8000多个字符，它们并不是很安全的，真正的处理要经过加密，直接使用第三方加密的库。

（4）小案例---服务器端进行渲染的（后端渲染）

数据正常是在数据库中的，现在用假数据来代替。

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>post测试</title>  </head>  <body>  <form action="**http://www.myWebServer.com/index.php**" method="**post**">  学号：<input type="text" name="stuId"><br />  <input type="submit" value="查询">  </form>  </body>  </html> |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>Document</title>  </head>  <body>  <?php  // header('content-Type:text/plain; charset=utf-8');有html结构就不需要了，已经有utf-8  // 这个页面的数据是直接在后端就进行渲染的，使用html标签来进行渲染  $mysql = array();  $mysql['110'] = array('name'=>'张三', 'chinese'=>100,'math'=>100,'english'=>100);  $mysql['111'] = array('name'=>'零四', 'chinese'=>101,'math'=>101,'english'=>101);  $mysql['112'] = array('name'=>'四十', 'chinese'=>102,'math'=>102,'english'=>102);  $mysql['113'] = array('name'=>'五五', 'chinese'=>103,'math'=>103,'english'=>103);  $stuId = $\_POST['stuId'];  if ($stuId === 'admin') {  foreach ($mysql as $value) {  echo "<ul><li>姓名：$value[name]</li><li>语文：$value[chinese]</li><li>数学：$value[math]</li><li>英语：$value[english]</li></ul>";  }  } else {  $stuIdData = $mysql[$stuId];  echo "<ul><li>姓名：$stuIdData[name]</li><li>语文：$stuIdData[chinese]</li><li>数学：$stuIdData[math]</li><li>英语：$stuIdData[english]</li></ul>";  }  ?>  </body>  </html> |

1. 表单form的相关知识

1、表单提交

submit() 方法触发 submit 事件，或规定当发生 submit 事件时运行的函数。$(selector).submit(function)。

使用 preventDefault() 函数来阻止对表单的提交。

ajaxSubmit()提交表单，使用第三方插件jquery.form实现；

通常情况下，我们直接通过form提交的话，提交后**当前页面跳转到form的action所指向的页面**。然而，很多时候我们比不希望提交表单后页面跳转，那么，我们就可以使用ajaxSubmit(obj)来提交数据。

这里涉及到表单的同步提交和异步提交。本文不赘述。

2、form表单的action 与url跳转是不一样的：

（1）form表单可以给后台传递数据，在后台直接可以用request对象去向前台请求数据。

form表单传递数据有两种方式：

method="post":这是传递大量数据时用的，在数据传递之前会先将数据打包，因此这种传递数据的方式会效率会比较慢，但是穿过的数据都能正确解析，因此传中文不会有乱码。

method="get"：以URL传递的，因为地址栏长度有限，所以对数据量是有限制的，而且传递的数据必须是ASCCI码值范围内的，因此，传中文会有乱码，需特殊处理。

（2）rl跳转到action只能通过url的参数来传递数据，action?参数名=参数值&……，在后台也可以用request对象去向前台请求url的参数值（url传中文参数需要进行特殊处理）

七、后台接口，一般需前端渲染

服务器端返回来的内容不一定是一个完整的页面，这个没有强制的要求，服务器端应该返回什么数据取决你的业务需求，你想通过url地址得到什么信息。

后端接口就是直接返回数据，我们在浏览器中通过url看到页面数据实际是服务器端返回过来的（目前是wamp中网站返回的）。

一般就是在url中加参数如flag，然后服务器中通过$\_GET[]获取参数来返回指定的内容。

|  |
| --- |
| 例子：后台接口  <?php  $data = $\_GET['flag'];  echo '得到的url中的flag值：'.$data.'<br />';  var\_dump($data);  if ($data === '1') {  echo 'ok';  }  ?>  当你的url的flag = 1时，服务器就返回ok给你，现在是直接输出在网页上。 |

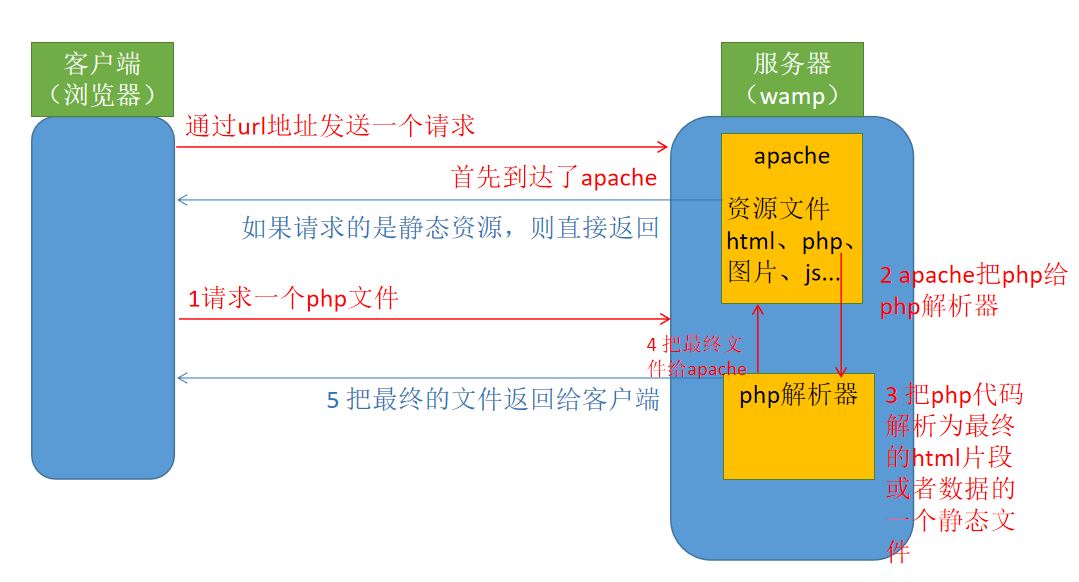
1. 后台渲染案例

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>前后端</title>  <style type="text/css">  table {  border-collapse: collapse;  }  </style>  </head>  <body>  <?php  $mysql = array();  $mysql[0] = array('name'=>'三国演义','author'=>'罗贯中','people'=>'战云','description'=>'战争');  $mysql[1] = array('name'=>'三国演义','author'=>'罗贯中','people'=>'战云','description'=>'战争');  $mysql[2] = array('name'=>'三国演义','author'=>'罗贯中','people'=>'战云','description'=>'战争');  $mysql[3] = array('name'=>'三国演义','author'=>'罗贯中','people'=>'战云','description'=>'战争');  ?>  <div class="wrapper">  <table border="1">  <thead>  <tr>  <th>书名</th>  <th>作者</th>  <th>人物</th>  <th>概述</th>  </tr>  </thead>  <tbody>  <?php  foreach ($mysql as $value) { **?>**  <tr>  <td><?php echo $value['name'] ?></td>  <td><?php echo $value['author'] ?></td>  <td><?php echo $value['people'] ?></td>  <td><?php echo $value['description'] ?></td>  </tr>  <?php }    ?>  </tbody>  </table>  </div>  </body>  </html> |

1. 请求的过程

如果请求的是一个html页面，那么现在跟wamp中的php是没有关系的，只使用了apache，它给我们提供了静态网页的服务。

如果请求的是一个php文件的话，它就不是静态资源，是动态内容，apache是不认识php的，它靠wamp中php去识别，然后整合到apache中。



在sublime，ctrl shift p 输入set 可以设置语法格式

九、Ajax

1、隐藏帧iframe方式实现局部更新

表单提交请求后，服务器把数据页面返回给ifame，iframe通过js操作父页面中的元素，让其显示数据，然后把iframe本身的width，height设置0，自己隐藏起来，这样不用等待服务器响应，完成局部更新。从表象看达成了异步的效果，本质上还不是异步。它还是同步的。

（1）iframe具体参数看w3c参考手册，可以理解为页面的子页面

|  |
| --- |
| <body>  <div class="wrapper">  <table border="1">  <thead>  <tr>  <th>书名</th>  <th>作者</th>  <th>人物</th>  <th>概述</th>  </tr>  </thead>  <tbody>  <tr>  <td>三国演义</td>  <td>罗贯中</td>  <td>赵云</td>  <td>战争</td>  </tr>  </tbody>  </table>  </div>  <iframe src="C:\Users\ASUS\Desktop\html\_study\workplace\Project\_practice\plane\index.html" width="600" height="600" frameborder="0"></iframe>  </body> |
|  |

1. form标签



（3）实例

测试时候发生了跨域的问题。

一开始html在本地，php在wamp服务器上。子页面的内容是服务器的，这样造成了跨域，无法直接访问。

把这两个文件都放wamp，同个域名下。通过域名访问就没有问题。但是通过C:\Users\ASUS\Desktop\html\_study\workplace\Ajax\myWebServer\xxx.html这样访问是不行的，它依然是本地的一个静态页面，得通过域名访问。

点击之后，实际上是刷新了，很快。

（4）iframe 子页面和父页面必须属于同一个域下

iframe 页面获取父页面的对象方法是 parent。

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>post测试</title>  <style type="text/css">  #login {  display: inline-block;  width: 100px;  background-color: rgba(0,0,0,.4);  text-align: center;  color: #fff;  margin-left: 30px;  border-radius: 10px;  }  </style>  </head>  <body>  <form action="index.php" method="post" target="myIframe">  账号：<input type="text" name="username"><span id="login"></span><br />  密码：<input type="text" name="password"><br />  <input type="submit" value="登陆" id="btn">  </form>  <iframe name="myIframe" width="0" height="0" frameborder="0" ></iframe>  </body>  </html> |
| <?php  **header('content-Type:text/html; charset=utf-8');// 根据实际来写形式**  $username = $\_POST['username'];  $pw = $\_POST['password'];  if ($username === 'admin' && $pw === '123') { ?>  <script type="text/javascript">  parent.document.getElementById('login').innerText = '登陆成功';  </script>  <?php } else { ?>  <script type="text/javascript">  parent.document.getElementById('login').innerText = '登陆失败';  </script>  <?php }  ?> |

2、原生Ajax详解-发送请求步骤

（1）创建XMLHttpRequest对象

早期的IE 6浏览器它是没有这个对象的，但是它有一个做同样功能的对象：ActiveXObject。

var xhr = new ActiveXobject(‘Microsoft.XMLHTTP’);参数是固定的。

（2）准备发送

open()

参数一：请求的方式（get获取数据；post提交数据）。

参数二：url地址，如果是get需要把参数拼接在url后边。”xxx.php?flag=1&flag1=2”

|  |
| --- |
| 1、get的方式  注：需要对中文进行处理  **encodeURI()**用来对中文参数进行编码，防止乱码。  xhr.open('get', encodeURI(url), true);  服务器端要用$\_GET[ ]去获取 |
| 1. post的方式   post请求参数在send()中传递，并且不需要 encodeURI() 转码，在发送前必须设置请求头，这个是固定的写法。  xhr.setRequestHeader(“Content-Type”,”application/x-www-form-urlencoded”);  Xhr.send(param);  服务器端要用$\_POST[ ]去获取 |

参数三：同步或者异步标志位，默认就是异步可以省略，true表示异步；false表示同步。

|  |
| --- |
| 使用同步的话，console控制台会提醒，这样做影响用户的体验。建议不这样做，这种应用的场景是非常少见的。 |

（3）执行发送动作

send();

如果是get，参数则是null，这个是必须传的，在早期的浏览器中，对它有一个要求，就是没有数据的时候，给它置空。

如果是post，则传入相应的数据。

（4）指定回调函数，最关键的一步

onreadystatechange = function () {...}。现在只是注册而已，没有执行。由这个函数来接收服务器端返回来的数据。

这个是浏览器帮我们去调用的。它会帮我们去监测服务器端返回的数据，该函数需要去进行状态的判断readyState = 4的时候，说明服务器数据已经到。 还得在判断数据是否正常，status = 200，说明服务器数据正常。再去接收数据，data = xhr.responseText; 接下来就是对数据的处理。

（5）回调函数中状态值分析

|  |  |
| --- | --- |
| xhr.readyState  0 xhr对象初始化。  1 执行发送动作，已经发送了请求  2 浏览器已经收到了服务器的响应的数据，此时还没有把数据解析出来，可能有包头，包尾，http传输时，会将数据先层层打包。  3 数据正在解析。  4 数据解析完成，可以使用了。但是这个数据不一定是正常的   |  | | --- | | xhr.state  200 数据相应正常。  404 没有找到资源。  500 服务器端错误。 |   xhr.reponseXML，正常的响应数据，早期用得比较多  xhr.reponseText，这个现在用得多，重点 |

该函数的调用条件是readyState状态发生变化（不包0、1），这里是浏览器自己去监测的。

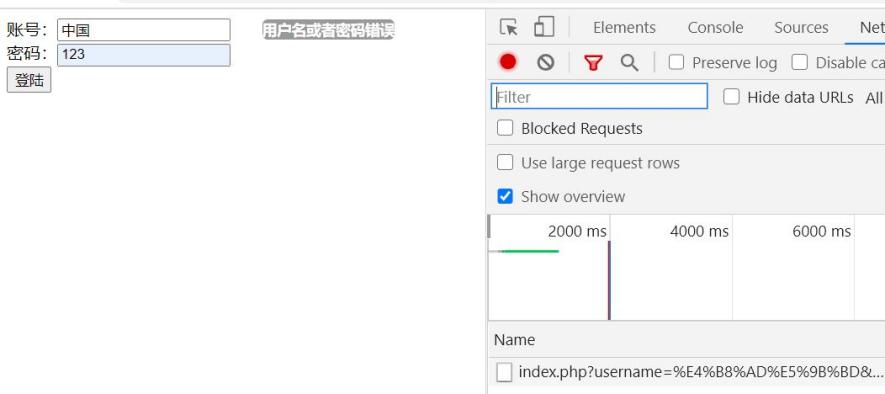
|  |
| --- |
| var xhr = null;  if (window.XMLHttpRequest) {  xhr = new XMLHttpRequest();  } else {  xhr = new ActiveXObject('Microsoft.XMLHTTP');  }  console.log(xhr.readyState +'------1号位');  // 2  var url = 'http://www.myWebServer.com/index.php';  var param = 'username='+username +'&password=' + password;  xhr.open('post', url, true);  // 3  xhr.setRequestHeader('Content-Type','application/x-www-form-urlencoded');  xhr.send(param);  // 4  console.log(xhr.readyState +'------2号位');  xhr.onreadystatechange = function () {  console.log(xhr.readyState +'------3号位');  if (xhr.readyState === 4) {  if (xhr.status === 200) {  var data = xhr.responseText;  alert(data);  }  }  }  2 |

3、实例，过这四步之后就是成功发送了一次ajax请求。

|  |
| --- |
| <?php  $username = $\_GET['username'];  $password = $\_GET['password'];  if ($username === 'admin' && $password === '123') {  echo '1';  } else {  echo '2';  }  ?> |
| **get的方式**  <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>post测试</title>  <style type="text/css">  #login {  display: inline-block;  background-color: rgba(0,0,0,.4);  text-align: center;  color: #fff;  font-weight: 900;  font-size: 14px;  margin-left: 30px;  border-radius: 5px;  }  </style>  </head>  <body>  <form>  账号：<input type="text" id="username"><span id="login"></span><br />  密码：<input type="text" id="password"><br />  <input type="button" value="登陆" id="btn">  </form>  <script type="text/javascript">  var btn = document.getElementById('btn');  btn.onclick = function () {  var username = document.getElementById('username').value;  var password = document.getElementById('password').value;  var url = 'http://www.myWebServer.com/index.php?username='+username +'&password=' + password;  // 1  // var xhr = new XMLHttpRequest();  var xhr = null;  if (window.XMLHttpRequest) {  xhr = new XMLHttpRequest();  } else {  xhr = new ActiveXObject('Microsoft.XMLHTTP');// IE6支持这个  }  // 2  // xhr.open('get', url, true);  xhr.open('get', encodeURI(url), true);// 增加对中文的编码，兼容IE浏览器  // 3  xhr.send(null);  // 4  xhr.onreadystatechange = function () {  if (xhr.readyState === 4) {  if (xhr.status === 200) {  var data = xhr.responseText;  if (data === '1') {  var login = document.getElementById('login');  login.innerText = '登陆成功';  } else {  var login = document.getElementById('login');  login.innerText = '用户名或者密码错误';  }  }  }  }  };  </script>  </body>  </html> |
| post方式请求，服务器记得通过 $\_POST方式去获取，不要还使用get方式获取，这样是获取不到的。  <form>  账号：<input type="text" id="username"><span id="login"></span><br />  密码：<input type="text" id="password"><br />  <input type="button" value="登陆" id="btn">  </form>  <script type="text/javascript">  var btn = document.getElementById('btn');  btn.onclick = function () {  var username = document.getElementById('username').value;  var password = document.getElementById('password').value;    // 1  // var xhr = new XMLHttpRequest();  var xhr = null;  if (window.XMLHttpRequest) {  xhr = new XMLHttpRequest();  } else {  xhr = new ActiveXObject('Microsoft.XMLHTTP');  }  // 2  var url = 'http://www.myWebServer.com/index.php';  var param = 'username='+username +'**&**password=' + password;  xhr.open(**'post'**, url, true);  // 3  xhr.**setRequestHeader**('Content-Type','application/x-www-form-urlencoded');  xhr.send(**param**);  // 4  xhr.**onreadystatechange** = function () {  if (xhr.readyState === 4) {  if (xhr.status === 200) {  var data = xhr.responseText;  alert(data);  }  }  }  };  </script> |
| 有问题可以看network中的preview，会有一些错误信息。 |

（1）传输中涉及汉字问题

它会对中文进行编码处理，服务器返回的数据能正常打印出来。





在IE浏览器会报错，需要对汉字进行处理，js中有处理的API可以使用

4、响应的数据格式

（1）XML（作为了解，目前很少用了，主流是json）

xml数据格式

元数据（标签）：用于描述数据的数据。

这种数据格式的弊端：

元数据占用的数量比较大，不利于大量数据的网络传输。

解析不太方便。

如果没有数据格式，那么在解析的时候是很麻烦的。xml在ajax返回的时候是一个类似于document对象文档。



|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>获取xml文件</title>  <style type="text/css">  #box {  width: 300px;  height: 300px;  background-color: gray;  color: #fff;  }  </style>  </head>  <body>  <input type="button" id="btn" value="获取xml">  <div id="box"></div>  <script type="text/javascript">  var btn = document.getElementById('btn');  btn.onclick = function () {  var xhr = null;  if (window.XMLHttpRequest) {  xhr = new XMLHttpRequest();  } else {  xhr = new ActiveXObject('Microsoft.XMLHTTP');  }  var url = 'http://www.myWebServer.com/data.xml';  xhr.open('get', encodeURI(url), true);  xhr.send(null);  xhr.onreadystatechange = function () {  if (xhr.readyState === 4) {  if (xhr.status === 200) {  var data = xhr.responseXML;  // 此时获取的是一个booklist的集合  var booklist = data.getElementsByTagName('booklist');  var book = booklist[0].children;  var name = book[0].children[0].textContent;  var author = book[0].children[1].textContent;  var desc = book[0].children[2].textContent;  // console.log(name, author, desc);  var str = name +' '+ author+' '+ desc;  var box = document.getElementById('box');  box.innerText = str;  }  }  }  }  </script>  </body>  </html> |
| data.xml  <?xml version="1.0" encoding="utf-8"?>  <booklist>  <book>  <name>三国演义</name>  <author>罗贯中</author>  <desc>一个站在纷争的时代</desc>  </book>  <book>  <name>西游记</name>  <author>吴承恩</author>  <desc>西天取经</desc>  </book>  </booklist> |
| php做xml的接口，index.php  使用php的话要指定header设置响应头的信息，保证浏览器可以把响应的内容识别为xml文件类型。  1  如果不设置，直接请求的话，浏览器通过xhr.responseXML得到的是null，因为浏览器认为数据是html格式的，所以不给responseXML赋值。  用浏览器去直接访问php文件，浏览器以html格式去解析php文件返回的数据。  1 |
| <?php  **header('Content-Type:text/xml;');**  $arr = array();  $arr[0] = array('name'=>'三国演义', 'author'=>'罗贯中', 'desc'=>'一个站在纷争的时代');  $arr[1] = array('name'=>'西游记', 'author'=>'吴承恩', 'desc'=>'西天取经');  ?>  <?xml version="1.0" encoding="utf-8" ?>  <booklist>  <book>  <name>**<?php echo** $arr[0]['name'] **?>**</name>  <author>**<?php echo** $arr[0]['author'] **?>**</author>  <desc>**<?php echo** $arr[0]['desc'] **?>**</desc>  </book>  <book>  <name><?php echo $arr[1]['name'] ?></name>  <author><?php echo $arr[1]['author'] ?></author>  <desc><?php echo $arr[1]['desc'] ?></desc>  </book>  </booklist> |

5、json数据格式

JavaScript 对象表示法（JavaScript Object Notation）。

JSON 是存储和交换文本信息的语法，类似 XML。

JSON 比 XML 更小、更快，更易解析。

（1）json数据和普通的js对象的区别

json数据没有变量

json形式的数据结尾没有分号

json数据中的键必须用双引号，不能使用单引号。

（2）JSON.parse()，把json形式的字符串转换成对象。

|  |
| --- |
| var str = '{"name":"hello","sex":boy}';  console.log(JSON.parse(str));  不符合json形式的话，直接报错。  1 |

（3）JSON.stringify()，把对象转换成json形式，返回转换后的字符串。

（4）eval()是作用就是把字符串解析成js代码并执行。加个括号是让它成为表达式。在早期用得比较多，但是有安全隐患，如果这个字符串中有js代码来操作本地的资源，可能会有危险，如获取账号密码。

注：拼接字符串，可以先把所有标签写好，然后在数据的位置加两个单引号，再加两个加号就可以了。

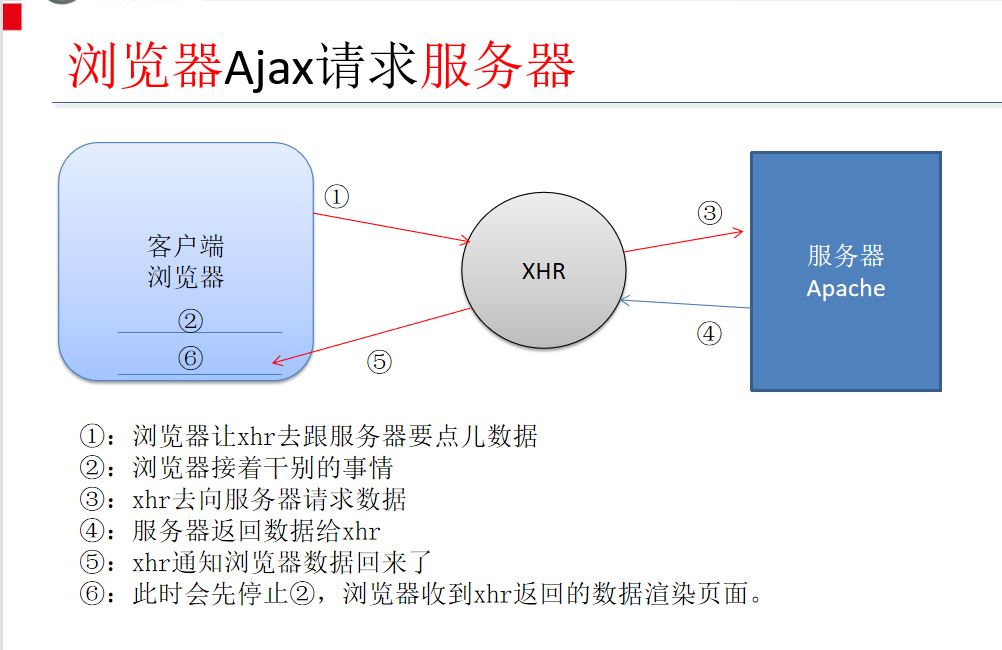
|  |
| --- |
| {  "name": "Mike",  "age": 32,  "hobbies": ["playing","dancing","swiming"],  "friend": {  "name": "Jon",  "age": 33,  "eat": "fruit"  }  }  一般情况下，数据是不会以这样的json文件格式，因为这样数据被写死了，实际是其他后台语言，获取数据库的数据，以这种json形式的数据返回回来。 |
| <input type="button" id="btn" value="获取xml">  <div id="box"></div>  <script type="text/javascript">  var btn = document.getElementById('btn');  btn.onclick = function () {  var xhr = null;  if (window.XMLHttpRequest) {  xhr = new XMLHttpRequest();  } else {  xhr = new ActiveXObject('Microsoft.XMLHTTP');  }  var url = 'http://www.myWebServer.com/data.json';  xhr.open('post', url, true);  xhr.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');  xhr.send('hello');  xhr.onreadystatechange = function () {  if (xhr.readyState === 4) {  if (xhr.status === 200) {  var data = xhr.responseText;  var d = **eval**('('+data+')');  // var d = **JSON.parse**(data);  var str = '<ul><li>'+d.name+'</li><li>'+d.age+'</li><li>'  + d.hobbies+'</li><li>'+ d.friend.eat+'</li></ul>';  var box = document.getElementById('box');  box.innerHTML = str;  // 把对象转换成json  var temp = **JSON.stringify**(d);  console.log(temp);  }  }  }  }  </script> |
| 1 |

1. php开发json接口
2. 数组也是属于json形式的范畴，可以理解成数字是json形式的一个特例。

在php中 json\_encode() 就是把数组转换成json形式的字符串。

|  |
| --- |
| <?php  // echo '{"name": "Mike", "age": "18", "sex": "male"}';    // $arr = array(1,2,3);  // $json = json\_encode($arr);  // echo $json;  $arr = array('name'=>'Mike','age'=>18,'sex'=>'male');  $jsonStr = json\_encode($arr);  echo $jsonStr;  ?> |

1. 浏览器Ajax请求服务器



十、异步效果与js事件处理机制

1、js的事件处理机制：单线程 + 事件队列

2、事件队列中的任务执行的条件：

（1）主线程已经空闲

（2）任务满足触发条件

|  |
| --- |
| 定时函数（延时时间已经达到）  事件函数（特定的事件被触发，如点击，鼠标经过...事件）  ajax的回调函数（服务器端有数据响应） |
| 1、定时函数  setTimeout(function () {...}, 0)  setInterval()  2、事件函数  btn.onclick = functoin () {...}  btn.onmouseenter = function () {...}  3、Ajax回调函数  xhr.onreadystatechange = function () {...} |

js的解析器执行到注册/绑定（不会执行）事件处理函数的时候，就把回调函数加入到事件队列中去。

当单线程(或者称为主线程，它只有一个线程)运行结束之后，浏览器会提供另外的循环机制---事件循环线程。

事件循环线程，就是判断任务是否满足触发条件，以及主线程是否空闲了。如果两者都满足了，则把任务（即是回调函数）给主线程去执行。

1. ajax初步封装

1、例子，两个按钮点击的时候，要根据post和get去解开和注释对应的index.php文件，把他们的后台接口直接搞一起。

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>ajax封装</title>  <style type="text/css">  #box {  width: 300px;  height: 300px;  background-color: gray;  color: #fff;  }  #box1 {  width: 300px;  height: 300px;  background-color: gray;  color: #fff;  }  </style>  </head>  <body>  <input type="button" id="btn" value="get获取index.php">  <div id="box"></div>  <input type="button" id="btn1" value="post获取index.php">  <div id="box1"></div>  <script type="text/javascript">  var btn = document.getElementById('btn');  btn.onclick = function () {  var url = 'http://www.myWebServer.com/index.php';  var type = 'get';  var parma = 'username=李四&password=123';  ajax(type, url, parma, function (data) {  var box = document.getElementById('box');  box.innerHTML = data;  });  }  var btn1 = document.getElementById('btn1');  btn1.onclick = function () {  var url = 'http://www.myWebServer.com/index.php';  var type = 'post';  var parma = 'username=lxf&password=123';  ajax(type, url, parma, function (data) {  var box1 = document.getElementById('box1');  data = JSON.parse(data);  var str = '<ul><li>'+data.username+'</li><li>'+data.password+'</li></ul>';  box1.innerHTML = str;  });  }  // parma在传递的时候，需要符合要求，现在没有出错的判断  /\*  如果没有传参数的话，怎么处理，以及参数的顺序是不能颠倒的。学jQuery的封装。  \*/  function ajax(method, url, parma, callback) {  var xhr = null;  if (window.XMLHttpRequest) {  xhr = new XMLHttpRequest();  } else {  xhr = new ActiveXObject('Microsoft.XMLHTTP');  }  if (method === 'get' && parma !== 'undefined') {  url += '?' + parma;  }  // url 加上 encodeURI() 是兼容IE6  xhr.open('get', url, true);  if (method === 'post') {  xhr.open('post', url, true);  xhr.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');  }  var data = null;  if (method === 'post') {  data = parma;  }  xhr.send(data);  xhr.onreadystatechange = function () {  if (xhr.readyState === 4) {  if (xhr.status === 200) {  var data = xhr.responseText;  callback(data);  }  }  }  }  </script>  </body>  </html> |
| <?php  /\*$username = $\_GET['username'];  $password = $\_GET['password'];  echo $username.'---------'.$password;\*/  $username = $\_POST['username'];  $password = $\_POST['password'];  echo '{"username": "'.$username.'", "password":"'.$password.'"}';  ?> |

2、查询书籍案例

（1）array\_key\_exists用来判断数组中有没有对应键，返回布尔类型

|  |
| --- |
| <?php  $bookName = $\_GET['bookName'];  $booklist = array();  $booklist['xyj'] = array('name'=>'西游记','author'=>'吴承恩','desc'=>'西天取经的故事');  $booklist['sgyy'] = array('name'=>'三国演义','author'=>'罗贯中','desc'=>'战争贩子');  $booklist['hlm'] = array('name'=>'红楼梦','author'=>'曹雪芹','desc'=>'一家人的故事');  $booklist['shz'] = array('name'=>'水浒传','author'=>'施耐庵','desc'=>'108好汉');  if (**array\_key\_exists**($bookName, $booklist) === true) {  echo '{"name":"'.$booklist[$bookName]['name'].'","author":"'.$booklist[$bookName]['author'].'","desc":"'.$booklist[$bookName]['desc'].'"}';  } else {  echo '{"flag": 0}';  }  ?> |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>ajax封装应用小案例</title>  <style type="text/css">  .container {  width: 300px;  height: 300px;  background-color: lightgreen;  }    </style>  </head>  <body>  <div class="container">  图书名字<input type="text" id='bookName'>  <input type="button" value="查询" id="btn">  <div id="show"></div>  </div>  <script type="text/javascript">  var btn = document.getElementById('btn');  btn.onclick = function () {  var bookName = document.getElementById('bookName').value;  var url = 'http://www.myWebServer.com/index.php';  var parma = 'bookName='+bookName;  var method = 'get';  ajax(method, url, parma, function (data) {  data = JSON.parse(data);  if (data.flag !== 0) {  var str = '<ul><li>书名：'+data.name+'</li><li>作者：'+data.author+'</li><li>描述：'+data.desc+'</li></ul>';  } else {  var str = '查询不到该图书';  }  var show = document.getElementById('show');  show.innerHTML = str;  });    };  // parma在传递的时候，需要符合要求，现在没有出错的判断  /\*  如果没有传参数的话，怎么处理。学jQuery的封装  \*/  function ajax(method, url, parma, callback) {  var xhr = null;  if (window.XMLHttpRequest) {  xhr = new XMLHttpRequest();  } else {  xhr = new ActiveXObject('Microsoft.XMLHTTP');  }  if (method === 'get' && parma !== 'undefined') {  url += '?' + parma;  }  // url 加上 encodeURI() 是兼容IE6  xhr.open('get', url, true);    if (method === 'post') {  xhr.open('post', url, true);  xhr.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');  }  var data = null;  if (method === 'post') {  data = parma;  }  xhr.send(data);  xhr.onreadystatechange = function () {  if (xhr.readyState === 4) {  if (xhr.status === 200) {  var data = xhr.responseText;  callback(data);  }  }  }  }  </script>  </body>  </html> |

十二、jQuery的Ajax

1、异步调用

dataType: ’json ’，其值可以是 xml、json、text、html、script、jsonp，实际只有xml，text其他是经过封装了，因为原生只有xhr.responseXML、xhr.responseText

error : function (data) {...出错的时候，可以给用户一些提示}

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>jQuery调用ajax</title>  <style type="text/css">  .container {  width: 300px;  height: 300px;  background-color: lightgreen;  }  </style>  </head>  <body>  <div class="container">  图书名字<input type="text" id='bookName'>  <input type="button" value="查询" id="btn">  <div id="show"></div>  </div>  <script type="text/javascript" src="jquery-1.12.2.js"></script>  <script type="text/javascript">  $(function () {  $('#btn').click(function () {  $.ajax({  url: 'http://www.myWebServer.com/index.php',  type: 'GET',  data: {bookName: $('#bookName').val()},  dataType: 'json',  success: function (data) {  var str = null;  if (data.flag === 0) {  str = '查询的书籍未找到';  } else {  str = '<ul><li>'+data.name+'</li><li>'+data.author+'</li><li>'+data.desc+'</li></ul>';  }  console.dir(data);  $('#show').html(str);  }  });  });  });  </script>  </body>  </html> |
| <?php  $bookName = $\_GET['bookName'];  $booklist = array();  $booklist['xyj'] = array('name'=>'西游记','author'=>'吴承恩','desc'=>'西天取经的故事');  $booklist['sgyy'] = array('name'=>'三国演义','author'=>'罗贯中','desc'=>'战争贩子');  $booklist['hlm'] = array('name'=>'红楼梦','author'=>'曹雪芹','desc'=>'一家人的故事');  $booklist['shz'] = array('name'=>'水浒传','author'=>'施耐庵','desc'=>'108好汉');  if (array\_key\_exists($bookName, $booklist) === true) {  echo '{"name":"'.$booklist[$bookName]['name'].'","author":"'.$booklist[$bookName]['author'].'","desc":"'.$booklist[$bookName]['desc'].'"}';  } else {  echo '{"flag": 0}';  }  ?> |

使用对象来做参数，它里边属性当作参数，可以不用顺序，这样就能处理有些参数没有传递的问题。

2、同步调用

如果是同步的话，它不会走回调函数。数据要自己进行处理。

jQuery返回的数据是经过封装了，思路基本是一样的

|  |
| --- |
| <script type="text/javascript">  $(function () {  $('#btn').click(function () {  var ret = $.ajax({  url: 'http://www.myWebServer.com/index.php',  type: 'POST',  data: {bookName: $('#bookName').val()},  dataType: 'json',  **async: false**  });  console.log(ret);  **ret = ret.responseJSON;**  if (ret.flag !== 0) {  var str = '<ul><li>书名：'+ret.name+'</li><li>作者：'+ret.author+'</li><li>描述：'+ret.desc+'</li></ul>';  } else {  str = '查询不到该图书';  }  var show = document.getElementById('show');  show.innerHTML = str;  });  });  1 |

3、模仿jQuery ajax封装

对大小写还没有进行封装，根据get或者post的方式去更改php接口文件

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>ajax封装应用小案例</title>  <style type="text/css">  .container {  width: 300px;  height: 300px;  background-color: lightgreen;  }    </style>  </head>  <body>  <div class="container">  图书名字<input type="text" id='bookName'>  <input type="button" value="查询" id="btn">  <div id="show"></div>  </div>  <script type="text/javascript">  var btn = document.getElementById('btn');  btn.onclick = function () {  var bookName = document.getElementById('bookName').value;  /\*ajax({  type: 'post',  dataType: 'text',  data: {'bookName': bookName},  url: 'http://www.myWebServer.com/index.php',  async: true,  success: function (data) {  if (data.flag !== 0) {  var str = '<ul><li>书名：'+data.name+'</li><li>作者：'+data.author+'</li><li>描述：'+data.desc+'</li></ul>';  } else {  var str = '查询不到该图书';  }  console.log(data);  var show = document.getElementById('show');  show.innerHTML = data;  }  });\*/  var ret = ajax({  type: 'get',  dataType: 'json',  data: {'bookName': bookName},  url: 'http://www.myWebServer.com/index.php',  async: false  });  console.log(ret);  if (ret.flag !== 0) {  var str = '<ul><li>书名：'+ret.name+'</li><li>作者：'+ret.author+'</li><li>描述：'+ret.desc+'</li></ul>';  } else {  str = '查询不到该图书';  }  var show = document.getElementById('show');  show.innerHTML = str;  };  function ajax(obj) {  var origin = {  type: 'get',  dataType: 'text',  data: {},  url: '',  async: true,  success: function (data) {  console.log(data);  }  }  // 获取参数值,没有传递的参数的使用默认值  for (var key in obj) {  origin[key] = obj[key];  }  // 处理data  var data = '';  for (var key in origin.data) {  data = data + key + '=' + origin.data[key] + '&';  }  data = data.substring(0, data.length - 1);  // 处理url  if (origin.type === 'get') {  origin.url = origin.url + '?' + data;  }  // 开始ajax  var xhr = null;  if (window.XMLHttpRequest) {  xhr = new XMLHttpRequest();  } else {  xhr = ActiveXObject('Microsoft.XMLHTTP');  }  // encodeURI兼容IE6  xhr.open(origin.type, encodeURI(origin.url), origin.async);  var parma = null;  if (origin.type === 'post') {  parma = data;  // post要设置请求头,请求头没有设置好，返回来的数据是有问题的  xhr.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');  }  xhr.send(parma);  // 如果是同步的情况的话，直接返回 responseText  if (origin.async === false) {  if (origin.dataType === 'json') {  return JSON.parse(xhr.responseText);  } else {  return xhr.responseText;  }  }  xhr.onreadystatechange = function () {  if (xhr.readyState === 4) {  if (xhr.status === 200) {  var retData = xhr.responseText;  if (origin.dataType === 'json') {  retData = JSON.parse(retData);  // retData = eval(''+retData+'');  }  origin.success(retData);  }  }  };  }  </script>  </body>  </html> |
| <?php  // $bookName = $\_POST['bookName'];  $bookName = $\_GET['bookName'];  $booklist = array();  $booklist['xyj'] = array('name'=>'西游记','author'=>'吴承恩','desc'=>'西天取经的故事');  $booklist['sgyy'] = array('name'=>'三国演义','author'=>'罗贯中','desc'=>'战争贩子');  $booklist['hlm'] = array('name'=>'红楼梦','author'=>'曹雪芹','desc'=>'一家人的故事');  $booklist['shz'] = array('name'=>'水浒传','author'=>'施耐庵','desc'=>'108好汉');  if (array\_key\_exists($bookName, $booklist) === true) {  echo '{"name":"'.$booklist[$bookName]['name'].'","author":"'.$booklist[$bookName]['author'].'","desc":"'.$booklist[$bookName]['desc'].'"}';  } else {  echo '{"flag": 0}';  }  ?> |

十三、跨域

1、Ajax跨域

（1）同源策略

同源策略是浏览器的一种安全策略，所谓同源指的是请求URL地址中的协议、域名和端口都相同，只要其中之一不相同就是跨域。

同源策略主要为了保证浏览器的安全性。

在同源策略下，浏览器不允许Ajax跨域获取服务器数据。

http://www.example.com/detail.html

|  |  |
| --- | --- |
| http://api.example.com/detail.html | 域名不同 |
| https//www.example.com/detail.html | 协议不同 |
| http://www.example.com:8080/detail.html | 端口不同 |
| http://api.example.com:8080/detail.html | 域名、端口不同 |
| https://api.example.com/detail.html | 协议、域名不同 |
| https://www.example.com:8080/detail.html | 端口、协议不同 |

2、跨域解决方案

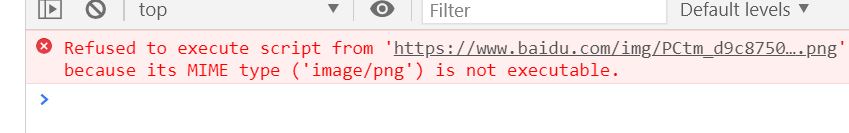
（1）jsonp---重点

主要采用这种方式来解决跨域的问题，主流的方案开发接口简单，无兼容性问题。

标签的src属性支持跨域请求，并且能够获取到服务器的数据。如img的src如果设置为百度图片的位置，能够访问获取到。

script标签直接引用php echo ‘hello’为什么出错，单纯输出数字1就没有问题。

答：script标签请求的会被当成js来进行解析，当用来请求一张图片的时候，会有如下的报错



（1）静态script标签的src属性进行跨域请求

就是通过script中的src来获取请求，请求的内容可以是一个js文件或者是一个php文件（php中返回来的实际上就一段js代码，这段代码就相当于是在script内声明了一个变量）。

script标签里面的**async属性**表示异步加载资源，默认情况下是同步加载。如果是异步的话，当你去访问变量的时候，它src请求的资源不一定加载完成了。一般是无法得到数据的，如果想通过这种方式来进行跨域，那么这个属性不能设置。

缺点：

（1）有顺序的问题，必须先加载后才能访问，就是请求的script要在前边。必须保证加载的顺序

（2）要想给src传参数的话，不太容易。不方便通过程序传递参数

静态的不常用。

（2）动态创建script标签，通过标签的src属性发送请求。（主流）

动态创建的script标签发出去的**请求是异步请求**，是不可以修改的，无法做成同步的。直接去获取数据是不行的。

|  |
| --- |
| <script type="text/javascript">  var script = document.createElement('script');  script.src = 'http://www.myWebServer.com/index.php';  var body = document.getElementsByTagName('body')[0];  body.appendChild(script);    **console.log(data);**  </script> |
| <?php  echo 'var data = 1';  ?> |
| 1 |

服务器可以**返回一个函数调用的形式**，一开始请求后，控制台会报错，没有这个函数，我们就可以给它先定义这个函数。

Jsonp的本质：通过jsonp来访问一个url地址，然后让它返回来的内容是一个函数调用，并且通过这个调用来传递一些参数。

这个函数的调用不用我们来显示的调用，服务器返回来的时候，浏览器帮我们去调用。

|  |
| --- |
| <body>  <script type="text/javascript">  var script = document.createElement('script');  script.**src** = 'http://www.myWebServer.com/index.php';  var body = document.getElementsByTagName('body')[0];  body.appendChild(script);  function foo(data) {  console.log(data);  }  // 这里的foo函数是由服务器响应的内容（js代码来调用的）  </script>    </body><!-- 相当于是这样子  <script type="text/javascript">  foo(arr);  </script> --> |
| <?php  $arr = array('name'=>'Mike', 'age'=>'18');  echo 'foo('.json\_encode($arr).')';  ?> |
| 1 |

存在的问题，返回的函数的函数名是固定，不灵活

在传参的时候 = 号前边的，主要用来后端获取数据用的，一般要根据后端，如php文件中的变量来命名；= 号后边的，前端用来定义这个函数叫什么名字。

（3）jsonp的本质：动态创建script标签，然后通过它的src属性发送跨域请求，然后服务器端响应的数据格式为“函数调用 foo(实参)”，所以在发送请求之前必须先声明一个函数，并且函数的名字与参数中传递的名字要一致，这里声明的函数是由服务器响应的内容（实际就是一段js代码---函数调用）来调用。

|  |
| --- |
| 这段代码相当是 jsonp 的实现。  <script type="text/javascript">  var script = document.createElement('script');  script.src = 'http://www.myWebServer.com/index.php?**callback**=**fnName**&username=admin&password=123';  var body = document.getElementsByTagName('body')[0];  body.appendChild(script);    function **fnName**(data) {  console.log(data);  }  </script>  如果 src 只是的值设置为http://www.myWebServer.com/index.php，那么它是无法获得数据 的，因为服务器需要username和password，如果没有传值过去，服务器没有获取到这两个参数，不知道要返回什么。所有无法返回。 |
| <?php  $**callback** = $\_GET[**'callback'**];  $username = $\_GET['username'];  $password = $\_GET['password'];  $arr = array('name'=>$username, 'password'=>$password);  echo $callback.'('.json\_encode($arr).')';  ?> |

（2）document.domain+iframe

（3）location.hash + iframe

（4）window.name + iframe

（5）window.postMessage

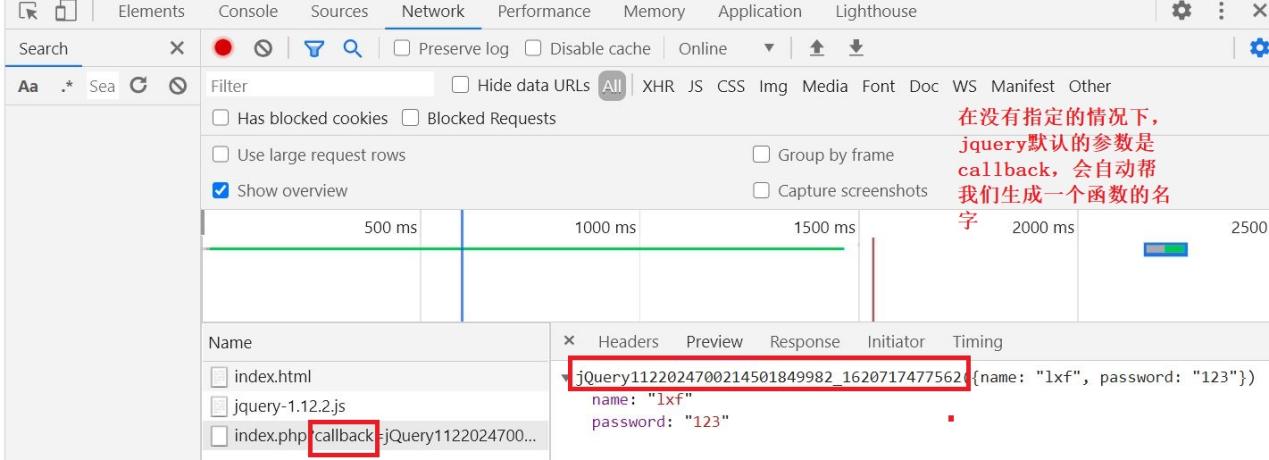
它是H5提供的API，有兼容性问题

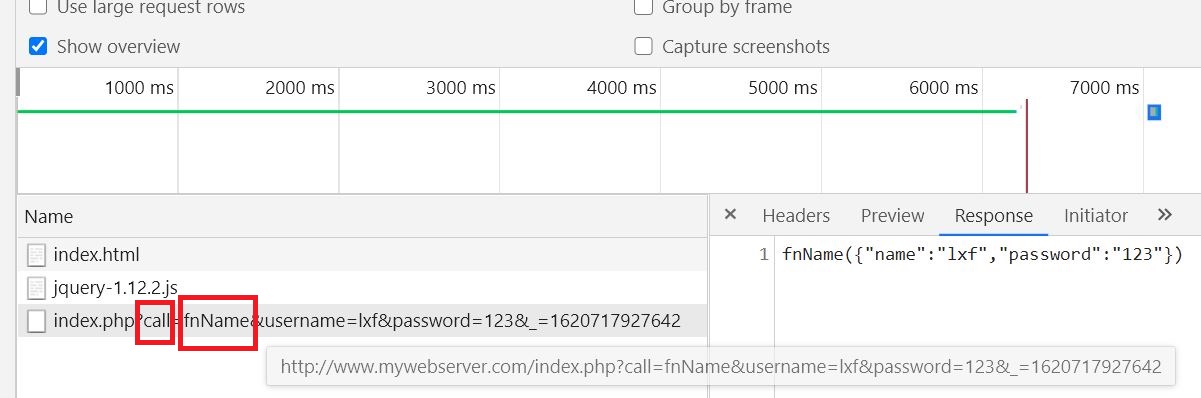
（6）flash等第三方插件

1. jQuery跨域的解决，对jsonp的支持

方法是封装在ajax中的，但是跟ajax没有什么关系，底层的实现方式完全不一样，它代码走的是上边的那种方式。

|  |
| --- |
| $(function () {  $('#btn').click(function () {  $.ajax({  url: 'http://www.myWebServer.com/index.php',  type: 'get',// 用post不行，它封装是get方式的，因为是通过src拼接请求的方式，通过url来请求  data: {username: 'lxf',password: '123'},  dataType: 'jsonp',  jsonp: 'call',// 要跟服务器一致，jsonp的属性的作用就是自定义参数名字（callback = abc 这里的名字指的是等号前面的键，后端根据这个键获取方法名，jQuery的默认参数是callback）。  jsonpCallback: 'fnName',// 这个属性的作用就是自定义回调函数的名字（callback = abc，这里的名字指的是等号后面的值）。  success: function (data) {  console.log(data);  },  error: function () {  console.log('error');  }  });  });  }); |
| $callback = $\_GET[**'call'**];  $username = $\_GET['username'];  $password = $\_GET['password'];  $arr = array('name'=>$username, 'password'=>$password);  echo $callback.'('.json\_encode($arr).')';  ?> |





（1）时间戳，jQuery在url后面加上下划线，这个叫时间戳，是为了防止早期的IE浏览器进行数据缓存，要是有数据变更它不会去请求，要防止数据缓存

4、模仿 jQuery-jsonp 封装

## 注：跨域的时候。

## jsonp: 'call',// 要跟服务器一致，jsonp的属性的作用就是自定义参数名字（callback = abc 这里的名字指的是等号前面的键，后端根据这个键获取方法名，jQuery的默认参数是callback），**两者都要改，不要改一边，忘了另外一边，它不会报错的**

|  |
| --- |
| <?php  // 要跟前端页面接口参数一致，不然出错误  $callback = $\_GET['cb'];  $username = $\_GET['username'];  $password = $\_GET['password'];  $arr = array('username'=>$username, 'password'=>$password);  echo $callback.'('.json\_encode($arr).')';  ?> |
| <?php  // $username = $\_GET['username'];  // $password = $\_GET['password'];  $username = $\_POST['username'];  $password = $\_POST['password'];  $arr = array('username'=>$username, 'password'=>$password);  echo json\_encode($arr);  ?> |

5、智能搜索案例

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>jQuery智能搜索案例</title>  <style type="text/css">  \* {  margin: 0;  padding: 0;  }  ul {  list-style: none;  }  .container {  width: 300px;  height: 300px;  background-color: lightgreen;  text-align: center;  padding: 10px;  }  .container ul {  background-color: lightgray;  }  .container ul li {  background-color: ;  text-align: left;  }  </style>  </head>  <body>  <div class="container">  搜索框：<input type="text" id="query">  <ul></ul>  </div>  <script type="text/javascript" src="jquery-1.12.2.js"></script>  <script type="text/javascript">  /\*https://www.baidu.com/sugrec?pre=1&p=3&ie=utf-8&json=1&prod=pc&from=pc\_web&sugsid=33797,33821,33848,33607&wd=java&req=2&csor=4&pwd=jav&cb=jQuery110203866182401155369\_1620737917596&\_=1620737917600 不能只是单单截取到sugrec这样无法获取数据\*/  $(function () {  $('#query').keyup(function () {  var kw = $(this).val();  $.ajax({  url:'https://sp0.baidu.com/5a1Fazu8AA54nxGko9WTAnF6hhy/su',  // url: 'https://www.baidu.com/sugrec?pre=1&p=3&ie=utf-8&json=1&prod=pc&from=pc\_web&sugsid=33797,33821,33848,33607',  // 这两个不同url返回的结果是不一样的  data: {wd: kw},  dataType: 'jsonp',  jsonp: 'cb',  jsonpCallback: 'getInfo',  success: function (data) {  console.log(data);  var strArr = ['<ul>'];  $.each(data.s, function (index, value) {  strArr.push('<li>'+ value +'</li>');  });  strArr.push('</ul>');  $('.container>ul').html(strArr.join(''));  },  error: function (data) {  console.log(data);  }  });  });  $('.container ul').on('mouseenter', 'li', function () {  $(this).css('backgroundColor', 'white').siblings().css({'backgroundColor':'lightgray'});  });  });    </script>  </body>  </html> |
| 2  **注：电脑需要联网才行** |

1. 快递查询小案例

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>jQuery智能搜索案例</title>  <style type="text/css">  \* {  margin: 0;  padding: 0;  }  ul {  list-style: none;  }  .container {  width: 400px;  height: 300px;  background-color: lightgreen;  text-align: center;  padding: 10px;  font-size: 10px;  }  .container li {  text-align: left;  margin-left: 50px;  }  </style>  </head>  <body>  <div class="container">  <input type="text" placeholder="请输入EMS快递单号" value="1065329952611" id="txt"><input type="button" id="btn" value="查询">  <ul id="show"></ul>  </div>  <script type="text/javascript" src="jquery-1.12.2.js"></script>  <script type="text/javascript">  /\*https://express.baidu.com/express/api/express?query\_from\_srcid=&isBaiduBoxApp=10002&isWisePc=10020&tokenV2=\_4Jvl0Km-MZKPifPaY8HFjlqwQojwYbn8N2M6fe2UiRwpMfH1EHgXaKG8wSgW4-1&cb=jQuery1102039162663904312245\_1620745960154&appid=4001&com=ems&nu=1065329952611&vcode=&token=&qid=a518f5ca00066eea&\_=1620745960159\*/  // 1065329952611  $(function () {  $('#btn').click(function () {  var kw = $(this).val();  $.ajax({  url:'https://express.baidu.com/express/api/express?query\_from\_srcid=&isBaiduBoxApp=10002&isWisePc=10020&tokenV2=\_4Jvl0Km-MZKPifPaY8HFjlqwQojwYbn8N2M6fe2UiRwpMfH1EHgXaKG8wSgW4-1&appid=4001&com=ems&qid=a518f5ca00066eea',  data: {nu: '1065329952611'},  dataType: 'jsonp',  jsonp: 'cb',  jsonpCallback: 'getInfo',  success: function (data) {  var context = data.data.info.context;  var arrStr = [];  $.each(context, function (index, value) {  var desc = value.desc;  var time = value.time;  var date = new Date();  date.setTime(time+'000');  date = date.getFullYear()+'年'+(date.getMonth()+1)+'月'+date.getDate()+'日'+(date.getHours() > 12? ' 下午':' 上午')+(date.getHours() >12 ? date.getHours() % 12 : date.getHours())+'时'+date.getMinutes()+'分'+date.getSeconds()+'秒';  arrStr.push('<li><div>'+date+'</div><div>'+desc+'</div></li>');  });  $('#show').html(arrStr.join(''));  },  error: function (data) {  console.log('error');  console.log(data);  }  });  });    });    </script>  </body>  </html> |
| 1 |

7、天气预报小案例

采用的是360的天气的接口，可以在360官网获取，查看script的请求，就能看到它的jsonp接口。

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>jQuery天气预报案例</title>  <style type="text/css">  \* {  margin: 0;  padding: 0;  }  ul {  list-style: none;  }  .container {  width: 400px;  min-height: 300px;  background-color: lightgreen;  text-align: center;  padding: 10px;  font-size: 10px;  }  .container .date {  text-align: left;  margin-left: 50px;  border: 1px solid #000;  margin-bottom: 10px;  font-weight: 900;  }  .container .date .day {  background-color: #fff;  }  .container .date .night {  background-color: skyblue;  }  </style>  </head>  <body>  <div class="container">  <select>  <option value="101281901">揭西</option>  <option value="101280601">深圳</option>  <option value="101280101">广州</option>  <option value="101281701">中山</option>  <option value="101300501">桂林</option>  </select>  <input type="button" id="btn" value="查询">  <div id="show">    </div>  </div>  <script type="text/javascript" src="jquery-1.12.2.js"></script>  <script type="text/javascript">  /\*https://cdn.weather.hao.360.cn/sed\_api\_weather\_info.php?code=101280601&param=pm25&v=1&app=hao360&\_jsonp=\_\_jsonp5\_\_&t=2701338\*/  $(function () {  $('#btn').click(function () {  var code = $('select').children(':selected').attr('value');  $.ajax({  url: 'https://cdn.weather.hao.360.cn/sed\_api\_weather\_info.php?',  data: {param: 'pm25', app: 'hao360', code: code},  jsonp: '\_jsonp',  jsonpCallback: 'getInfo',  dataType: 'jsonp',  success: function (data) {  // console.log(data);  var weather = data.weather;  var info = [];  var str = '';  $.each(weather, function(index, value){  var date = value.date;  var day = value.info.day;  var night = value.info.night;  // console.log(date, day, night);  str = '<div class="date">日期:'+date+  '<ul class="day">'+  '<li>白天天气:'+day[1]+'</li>'+  '<li>白天温度:'+day[2]+'</li>'+  '<li>白天风向:'+day[3]+'</li>'+  '<li>白天风速:'+day[4]+'</li>'+  '</ul>'+  '<ul class="night">'+  '<li>夜间天气:'+night[1]+'</li>'+  '<li>夜间温度:'+night[2]+'</li>'+  '<li>夜间风向:'+night[3]+'</li>'+  '<li>夜间风速:'+night[4]+'</li>'+  '</ul>'+  '</div>'  info.push(str);  });  $('#show').html(info.join(''));  }  });  });  });    </script>  </body>  </html> |
| 1 |

十四、模板引擎

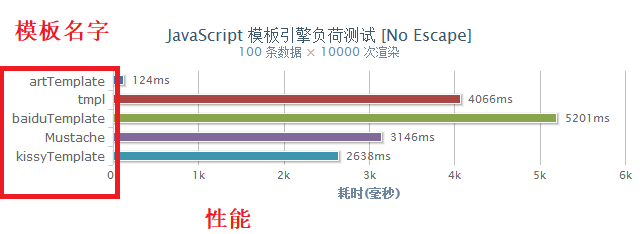
1、为何会有模板引擎

它跟数据解析有关系，我们在处理请求获得的数据，拼接字符串的时候，大家的做法都不一样，这样后期维护起来会有难度。

模板引擎就是规范数据的解析过程，它提供了一套规范的语法，维护起来会容易点。

2、模板+数据===>静态页面片段

流行的模板引擎



artTemplate是腾讯做到

kissyTemplate是阿里的

主要是学习循环和判断的语法

1、初步使用，这种用的比较多

template的作用就是把模板和数据拼接到一块生成一个静态的HTML片段，实际就是该方法的返回值。

参数一：模板id

参数二：用来渲染的数据

|  |
| --- |
| <div id="container"></div>  <script type="text/javascript" src="template-web.js"></script>  **// 1、先写好的模板**  <script type="text/html" id="template">  <h1>{{title}}</h1>  {{if type}}  {{each type as value i}}  <div>{{value}}</div>  {{/each}}  {{/if}}  </script>  <script type="text/javascript">  // function success(data) {  // // 模仿数据的处理  // var title = '<h2>'+ data.title +'</h2>';  // var str = '';  // for (var i = 0, len = data.type.length; i < len; i++) {  // str += '<div>'+ data.type[i] +'</div>';  // }  // var container = document.getElementById('container');  // container.innerHTML = title + str;  // }  var data = {  title: '水果',  type: ['西瓜','苹果','橙子','葡萄']  };  // success(data);  **// 2、调用模板**  var html = template('template', data);  var container = document.getElementById('container');  container.innerHTML = html;  </script>  **as在新版本中已经没有用了**  1 |

2、compile()和render()

如果模板比较小，那么就可以采用这种方式

|  |
| --- |
| <!DOCTYPE HTML>  <html>  <head>  <meta charset="UTF-8">  <title>compile-demo</title>  <script src="template-web.js"></script>  </head>  <body>  <h1>在javascript中存放模板</h1>  <div id="content"></div>  <script>  var source = '<ul>'  + '{{each list value i}}'  + '<li>索引 {{i + 1}} ：{{value}}</li>'  + '{{/each}}'  + '</ul>';  // compile用这个模板字符串编译出一个新的函数，渲染函数  var **render** = template.**compile**(source);  // render的作用就是用数据渲染静态标签内容  var html = **render**({  list: ['摄影', '电影', '民谣', '旅行', '吉他']  });  document.getElementById('content').innerHTML = html;  </script>  </body>  </html> |

3、模板引擎--天气预报案例

1. 综合案例

（1）省市县三级联动

（2）查询第三方天气信息（前后端综合）

1、通过后台获取第三方接口数据并封装为json接口

2、前端通过ajax方式调用封装好的后台接口

3、前端需要发送两次请求（体会异步编程模式）

如果想跨域获取json数据，可以通过同域的php后台文件去帮我们获取第三方数据，然后我们间接获得json数据。

如果第二次的ajax请求依赖于第一次请求的结果，那么第二次请求必须放到第一次回调函数的内部。

嵌入子模板实际上用的不多

转义跟不转义是怎么理解？？？

有的接口是可能是做了限制的，例如要在某个域名下才能去访问

每个城市都有编码的

在跨域的情况下，这个需求也是比较常见的，一个项目可能会分模块部署到不同的服务器上，如获取天气信息，地图信息，等等资源。

Ajax本身是解决不了的

什么是mime-type

简单来讲，它的作用就是服务器告诉浏览器你这个内容到底是个什么东东，是一张网页？还是一张图片？还是一个视频？浏览器只有知道了你这是个什么东西，才能正确处理它，mime就是这个东西，服务器通过Content-Type这个header来指定mime。