# 一 数据交互方式汇总

1. 表单 最基本的交互方式，http数据请求其实都是表单
2. ajax 不需要刷新就可以请求到数据结果，ajax也可以实现跨域（麻烦）
3. jsonp 跨域的实现方式之一，但是安全性很差，逐步被淘汰
4. WebSocket 与上述交互方式最大不同是双向的（双工），且性能很高、默认能跨域

单向：只能客户端请求服务端，如果服务端数据发生变化无法通知客户端也随之变化。ajax可以通过轮询方式实现双向通信，但是这样极度加重了服务器负担。

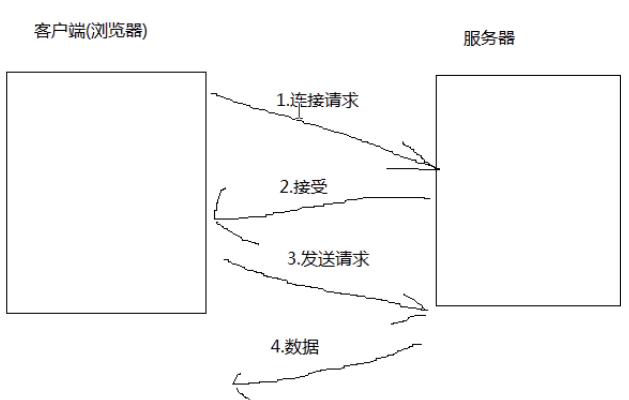
# 二 通信

http1.0 一次性的连接，已经淘汰

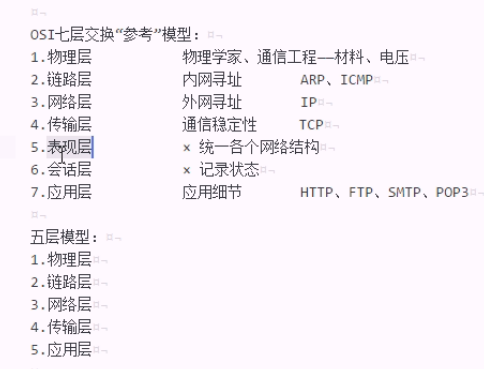
http1.1 保持连接，且性能提高

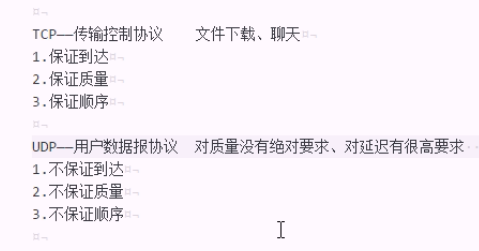
http2.0(草案) 强制https，自带双向通信，支持多路复用：一个服务传递多个数据

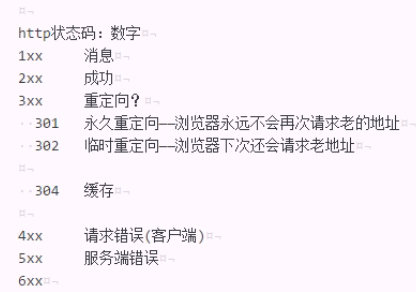
三次握手：



由图得知：连接数会直接影响服务器性能。



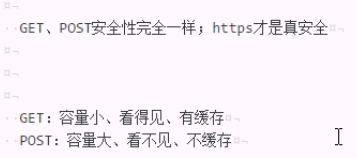




6以上都是扩展

# 三 数据提交

## 3.1 表单



<form action="\*\*.jsp" method="GET" id="form1">  
 账号：<input type="text" name="username" value=""><br>  
 密码：<input type="submit" name="password">  
 <input type="submit" value="提交">  
</form>  
<script>  
 *//前台校验* const $ = document.querySelectorAll.bind(document);  
 let oForm = $('#form1')[0];  
 let username = document.getElementsByName('username');  
 oForm.onsubmit = function () {  
 if(username.value == ''){  
 alert('用户名不能为空');  
 return false;  
 }  
 }

前台form表单提交POST的三种方式：



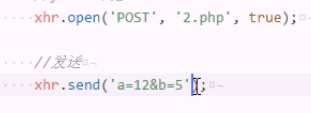
## 3.2 ajax

*//原生ajax*let btn = document.getElementById('btn');  
btn.onclick = function () {  
  
 *//第一步：创建xhr对象* if(window.XMLHttpRequest){  
 var xhr = new XMLHttpRequest();  
 } else {  
 var xhr = new ActiveXObject('Microsoft.XMLHttp'); *//兼容IE6* }  
  
 *//第二步：链接，默认参数为true--异步，false为同步，  
 // 注意：false时，浏览器会警告，因为同步操作基本被浏览器废弃* xhr.open('GET','http://localhost:8081/',true);  
  
 *//第三步：发送* xhr.send();  
 *//第四步：接收状态* xhr.onreadystatechange = function () {  
 /\*  
 readyState  
 0 初始化-刚刚创建  
 1 已连接  
 2 已发送  
 3 已接收-头信息  
 4 已接收-body信息  
 sta  
 \*/  
 if(xhr.readyState == 4){  
 if(xhr.status >= 200 && xhr.status < 300 || xhr.status == 304){  
 console.log('请求成功' + xhr.responseText);  
 }else {  
 console.log('请求失败');  
 }  
 }  
 /\*  
 ajax请求只有两种结果：  
 xhr.responseText 文本  
 xhr.responseXML XML  
 如果返回的数据是json，需要解析，在高级浏览器中提供了JSON对象解析  
 \*/  
 }  
}

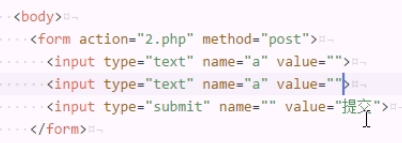
## 3.3 ajax与表单在post上的区别

如果使用post提交数据，其他设置都为默认，如

ajax：



form：



提交一些数据给后台，返回计算结果，则ajax会无法获取值，form却能获取，

因为这时候，同为POST请求，ajax默认的请求头 Conten-type 为’text/pain’

而form默认为

所以正确的ajax请求方式：



注意上述代码的顺序，不能随意调换。

常见的提交方式：

text/plain 纯文本 用不着

application/x-www-form-urlencoded &&&的方式 简单数据

multipart/form-data 定界符分割各个数据(文件上传) 上传文件

## 3.4 封装ajax

function ajax(options){  
 options=options||{};  
  
 options.type=options.type||'get';  
 options.data=options.data||{};  
 options.dataType=options.dataType||'text';  
  
 *//不兼容IE6* let xhr=new XMLHttpRequest();  
  
 *//数据组装* let arr=[];  
 for(let name in options.data){  
 arr.push(`${encodeURIComponent(name)}=${encodeURIComponent(options.data[name])}`);  
 }  
 let strData=arr.join('&');  
  
 if(options.type=='post'){  
 xhr.open('POST', options.url, true);  
 xhr.setRequestHeader('content-type', 'application/x-www-form-urlencoded');  
 xhr.send(strData);  
 }else{  
 xhr.open('GET', options.url+'?'+strData, true);  
 xhr.send();  
 }  
  
 *//接收* xhr.onreadystatechange=function (){  
 *//4——完事* if(xhr.readyState==4){  
 *//成功——2xx、304* if(xhr.status>=200 && xhr.status<300 || xhr.status==304){  
 let data=xhr.responseText;  
  
 switch(options.dataType){  
 case 'json':  
 if(window.JSON && JSON.parse){  
 data=JSON.parse(data);  
 }else{  
 data=eval('('+str+')');  
 }  
 break;  
 case 'xml':  
 data=xhr.responseXML;  
 break;  
 }  
  
 options.success && options.success(data);  
 }else{  
 options.error && options.error();  
 }  
 }  
 };  
}

## 3.5 jsonp

基础示例：

<script>  
 function show(json){  
 alert(json.s);  
 }  
</script>  
<script src="https://sp0.baidu.com/5a1Fazu8AA54nxGko9WTAnF6hhy/su?wd=yy&cb=show" charset="utf-8"></script>

即提前准备一个函数，该函数名与跨域请求对象的一个键值一样，即请求百度后，等待百度来调用我们自己的函数。

<body>  
<input type="text" id="txt">  
<script id="s1" charset="utf-8"></script>  
<script>  
 function show(json) {  
 alert(json.s);  
 }  
  
 let oS = document.getElementById('s1');  
 let oTxt = document.getElementById('txt');  
  
 oTxt.oninput = function () {  
 oS.src = `https://sp0.baidu.com/5a1Fazu8AA54nxGko9WTAnF6hhy/su?wd=${oTxt.value}&cb=show`;  
 };  
</script>  
  
</body>

但是script标签是一次性的。

标准的做法：

<input type="text" id="txt1">  
<ul id="ul1"></ul>  
<script>  
 function show(json) {  
 let oUl = document.getElementById('ul1');  
  
 oUl.innerHTML = '';  
 json.s.forEach(str => {  
 let oLi = document.createElement('li');  
  
 oLi.innerHTML = str;  
  
 oUl.appendChild(oLi);  
 });  
 }  
  
 let oTxt = document.getElementById('txt1');  
  
 oTxt.oninput = function () {  
 let oS = document.createElement('script');  
 oS.src = `https://sp0.baidu.com/5a1Fazu8AA54nxGko9WTAnF6hhy/su?wd=${encodeURIComponent(oTxt.value)}&cb=show`;  
  
 document.head.appendChild(oS);  
 };  
</script>