

**HTTP服务&AJAX编程**

目录

[目录 2](#_Toc455212326)

[第1章 服务器 6](#_Toc455212327)

[1. 服务器类型 6](#_Toc455212328)

[2. 服务器软件 6](#_Toc455212329)

[3. HTTP服务器 7](#_Toc455212330)

[第2章 客户端 7](#_Toc455212331)

[第3章 网络基础 7](#_Toc455212332)

[3.1 IP地址 7](#_Toc455212333)

[3.2 域名 8](#_Toc455212334)

[3.3 DNS服务 8](#_Toc455212335)

[3.4 端口 8](#_Toc455212336)

[第4章 C/S和B/S 8](#_Toc455212337)

[4.1 C/S结构 8](#_Toc455212338)

[4.2 B/S结构 9](#_Toc455212339)

[第5章 搭建HTTP服务 9](#_Toc455212340)

[5.1 安装WampServer 10](#_Toc455212341)

[5.2 管理HTTP服务 10](#_Toc455212342)

[5.3 配置根目录 11](#_Toc455212343)

[5.4 网站部署 12](#_Toc455212344)

[5.5 配置虚拟主机 12](#_Toc455212345)

[第6章 PHP基础 13](#_Toc455212346)

[6.1 入门 13](#_Toc455212347)

[6.2 变量 14](#_Toc455212348)

[6.3 数据类型 14](#_Toc455212349)

[6.4 内容输出 14](#_Toc455212350)

[6.5 运算符 15](#_Toc455212351)

[6.6 函数 15](#_Toc455212352)

[6.7 分支、循环语句 15](#_Toc455212353)

[6.8 表单处理 15](#_Toc455212354)

[6.9 常用PHP函数 16](#_Toc455212355)

[6.10 应用实例 17](#_Toc455212356)

[第7章 网络传输协议 17](#_Toc455212357)

[7.1 常见协议 17](#_Toc455212358)

[7.2 HTTP协议 17](#_Toc455212359)

[7.2.1 请求/请求报文 18](#_Toc455212360)

[7.2.2 响应/响应报文 19](#_Toc455212361)

[7.2.3 调试工具 20](#_Toc455212362)

[第8章 AJAX编程 21](#_Toc455212363)

[8.1 异步 21](#_Toc455212364)

[8.2 XMLHttpRequest 22](#_Toc455212365)

[8.2.1 请求 22](#_Toc455212366)

[8.2.2 响应 23](#_Toc455212367)

[8.2.3 API详解 24](#_Toc455212368)

[8.3 XML 26](#_Toc455212369)

[8.3.1 语法规则 26](#_Toc455212370)

[8.3.2 XML示例 26](#_Toc455212371)

[8.4 JSON 27](#_Toc455212372)

[8.4.1 语法规则 27](#_Toc455212373)

[8.4.2 JSON解析 27](#_Toc455212374)

[8.5 兼容性 28](#_Toc455212375)

[8.6 封装AJAX工具函数 28](#_Toc455212376)

[8.7 jQuery中的Ajax 28](#_Toc455212377)

[8.8 案例练习 30](#_Toc455212378)

[第9章 模板引擎 30](#_Toc455212379)

[9.1 原理剖析 30](#_Toc455212380)

[9.2 流行模板引擎 31](#_Toc455212381)

[9.3 artTemplate 31](#_Toc455212382)

[9.4 案例练习 31](#_Toc455212383)

[第10章 同源&跨域 31](#_Toc455212384)

[10.1 同源策略 31](#_Toc455212385)

[10.2 跨域 32](#_Toc455212386)

[10.3 跨域解决方案 32](#_Toc455212387)

[10.4 JSONP 33](#_Toc455212388)

[10.5 jQuery中的JSONP 35](#_Toc455212389)

[第11章 XMLHttpRequest 2.0 36](#_Toc455212390)

[11.1 设置超时 36](#_Toc455212391)

[11.2 FormData 36](#_Toc455212392)

[11.3 二进制 37](#_Toc455212393)

[11.4 上传进度 38](#_Toc455212394)

[第12章 综合练习 38](#_Toc455212395)

[12.1 瀑布流布局 38](#_Toc455212396)

# 服务器

通俗的讲，能够提供某种服务的机器（计算机）称为服务器。

## 服务器类型

按照不同的划分标准，服务可划分为以下类型：

1、按服务类型可分为：文件服务器、数据库服务器、邮件服务器、Web 服务器等；

2、按操作系统可分为：Linux服务器、Windows服务器等；

3、按应用软件可分为 Apache服务器、Nginx 服务器、IIS服务器、Tomcat服务器、Node服务器等。

## 服务器软件

使计算机具备提供某种服务能力的应用软件，称为服务器软件，通过安装相应的服务软件，然后进行配置后就可以使计算具备了提供某种服务的能力。

常见的服务器软件有：

1、文件服务器：Server-U、FileZilla、VsFTP等；

2、数据库服务器：Oracle、MySQL、PostgreSQL、MSSQL等；

3、邮件服务器：Postfix、Sendmail等；

4、HTTP 服务器：Apache、Nginx、IIS、Tomcat、NodeJS等；

## HTTP服务器

即网站服务器，主要提供文档(文本、图片、视频、音频)浏览服务，一般安装Apache、Nginx服务器软件。

HTTP服务器可以结合某一编程语言处理业务逻辑，由此进行的开发，通常称之为服务端开发。

常见的运行在服务端的编程语言包括 PHP、Jsp、Asp、Python、Ruby、Perl等。

# 客户端

具有向服务器索取服务能力的终端，如比如 手机、电脑等，通过安装不同的客户端软件，可以获取不同的服务，比如通过QQ获得即时通讯服务、通过迅雷获得下载服务等。

常见的客户端软件：浏览器、QQ、迅雷、Foxmail等。

以浏览器为宿主环境，结合 HTML、CSS、Javascript等技术，而进行的一系列开发，通常称之为前端开发。

# 网络基础

## IP地址

所谓IP地址就是给每个连接在互联网上的主机分配的一个32位地址。(就像每部手机能正常通话需要一个号码一样)

查看本机IP地址 ping、ipconfig、ifconfig

## 域名

由于IP地址基于数字，不方便记忆，于是便用域名来代替IP地址，域名是一个IP地址的“面具”

查看域名对应的IP地址 ping、tracert

## DNS服务

DNS记录了 IP 地址和域名的映射（对应）关系；

查找优先级 本机hosts文件、DNS服务器

## 端口

端口号是计算机与外界通讯交流的出口，每个端口对应不同的服务。

现实生活中，银行不同的窗口办理不同的业务。

查看端口占用情况 netstat -an

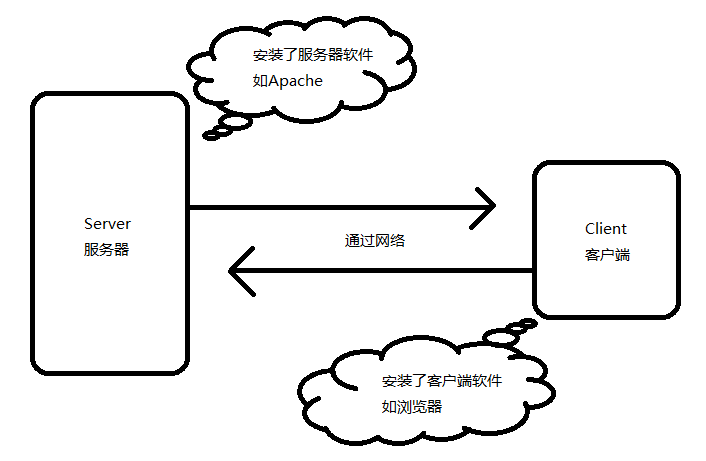
常见端口号 80、8080、3306、21、22

# C/S和B/S

## C/S结构

即Client、Server

C/S工作流程图



在C/S结构的情况下，不同的服务需要安装不同的客户端软件，比如QQ、迅雷、Foxmail这种情况下安装的软件会越来越多，同时也有许多弊端，比如A出差，需要在B电脑上查收邮件，但是B电脑并未安装Foxmail等类似的客户端软件，这样不得不先去下载Foxmail，非常不方便。

## B/S结构

B/S（即Broswer、Server）解决了C/S所带来的不便，将所有的服务都可以通过浏览器来完成（因为基本所有浏览器都安装了浏览器），但B/S也有一些不利，比如操作稳定性、流畅度等方面相对较弱。

# 搭建HTTP服务

Windows (Linux) + Apache + Mysql + PHP，首字母组合。

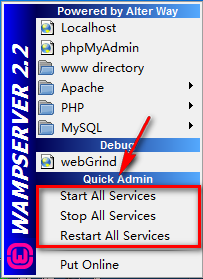
## 安装WampServer

安装wampserver，和普通软件安装无差别，除指定安装路径外，其它默认安装。

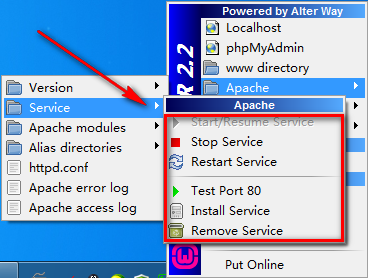
## 管理HTTP服务

任务图标绿色为正常启动状态

通过图形控制台可以启动、重启、停止所有服务



或者单独启动、重启、停止特定服务



**注意事项：**

1、检查网络是不是通的 ping 对方IP

2、检查防火墙是否开启，如果开启将不能正常被访问

3、检查访问权限 Allow from all

4、理解默认索引

5、确保端口没有被其它程序占用

6、“#”表示注释

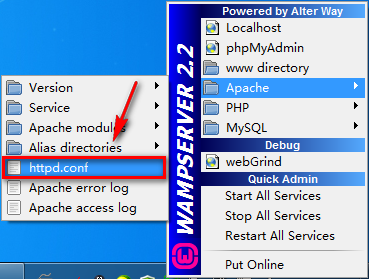
7、修改配置要格外小心，禁止无意修改其它内容

## 配置根目录

网站根目录是Web服务器上存放网站程序的空间，可通过修改配置文件自定义，如E:/www

具体步骤如下

1、打开配置文件，控制台选择



或者 wampserver安装目录下

bin\apache\Apache2.2.21\conf\httpd.conf

2、设定根目录，查找并修改



例如：



这样就指定了 "E:/www/"为存放网站的根目录。

3、配置根目录，查找



修改成



4、修改完后，并不能立即生效，需要重启Apache

注：可以指定任意目录为根目录

## 网站部署

将我们制作好的网页拷贝到配置好的根目录下，浏览器访问127.0.0.1即可。

## 配置虚拟主机

在一台Web服务器上，我们可以通过配置虚拟主机，然后分别设定根目录，实现对多个网站的管理。

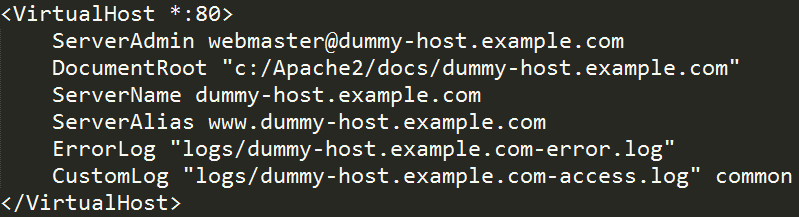
具体步骤如下：

1、开启虚拟主机辅配置，在httpd.conf 中找到



去掉前面的#号注释，开启虚拟主机配置

2、配置虚拟主机，打开conf/extra/httpd-vhosts.conf



分别修改以下三项

DocumentRoot "E:/www/example"

ServerName "example.com "

ServerAlias "www.example.com"

其它项无需指定。

3、修改DNS（hosts）文件

打开C:\Windows\System32\drivers\etc\hosts

这个目录是固定的



注：修改hosts文件权限

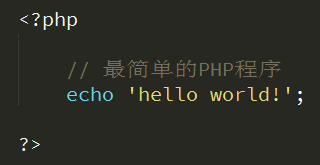
4、重启Apache

5、浏览器访问www.example.com

# PHP基础

## 入门

文件以.php后缀结尾，所有程序包含在<?php /\*\* 这里是代码 \*\*/ ?>



**注：避免使用中文目录和中文文件名**

见代码示例6-1.php

## 变量

1、变量以$开头 字母/数字/下划线 不能以数字开头

2、大小写敏感（区分大小写）

见代码示例6-2.php

## 数据类型

字符型、整型、浮点型、布尔型、数组、对象、NULL

单引号&双引号区别

索引数组、关联数组（了解即可）

见代码示例6-3.php

## 内容输出

echo：输出简单数据类型，如字符串、数值

print\_r()：输出复杂数据类型，如数组

var\_dump()：输出详细信息，如对象、数组（了解）

见代码示例6-4.php

## 运算符

基本与Javascript语法一致

. 号表示字符串拼接符，Javascript中为+号

见代码示例6-5.php

## 函数

与Javascript基本一致

函数名对大小写不敏感

默认参数（了解即可）

见代码示例6-6.php

## 分支、循环语句

与Javascript基本一致

foreach()

见代码示例6-7.php

## 表单处理

**HTML表单相关属性：**

|  |  |  |
| --- | --- | --- |
| **属性** | **示例** | **含义** |
| **action** | <form action="xx.php"> | 设置接收数据的处理程序 |
| **method** | <form method="post/get"> | 设置提交数据的方式 |
| **name** | <input name="username"> | 设置服务端接收数据的key |
| **enctype** | <form enctype="multipart/form-data"> | 设置提交数据编码 |

注：当上传文件是需要设置 enctype="multipart/form-data"，且只能post方式。

**PHP接收数据相关方法：**

|  |  |  |
| --- | --- | --- |
| **方法** | **使用** | **含义** |
| **$\_GET** | $\_GET['username'] | 接收以get形式提交的数据 |
| **$\_POST** | $\_POST['password'] | 接收以post形式提交的数据 |
| **$\_FILES** | $\_FILES['upload'] | 接收上传的文件 |

PHP接收的数据都是以数组方式，我们通过form表单里各个表单项指定的name属性值可以获得相应的提交数据，其中$\_FILES获得到的是一个二维数组，数组里包含了上传文件的相关信息。

见代码示例6-2.html和6-12.php

见代码示例6-3.html和6-13.php

## 常用PHP函数

in\_array() 是否在数组中

count() 计算数组长度

array\_key\_exists ()检测数组中是否存在key

file\_get\_contents读取文件

move\_uploaded\_file转存上传上来的文件

...还有很多

## 应用实例

1、用户登录

见代码示例login.html和login.php

2、动态网站

见代码示例ailvxing/index.html

# 网络传输协议

指服务器和客户端间进行通信时的约束和规范，客户端与服务端的数据交互并不是杂乱无章的，需要遵照（基于）一定的规范进行。

## 常见协议

1、HTTP、HTTPS 超文本传输协议

2、FTP 文件传输协议

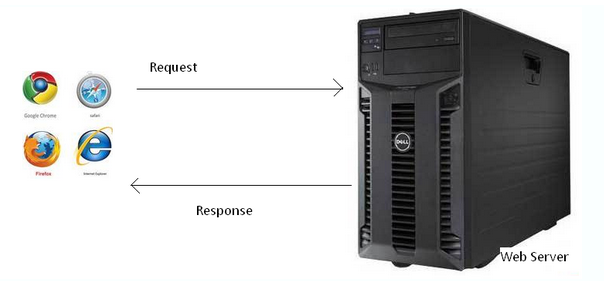
3、SMTP 简单邮件传输协议

## HTTP协议

即超文本传输协议，网站是基于HTTP协议的，例如网站的图片、CSS、JS等都是基于HTTP协议进行传输的。

HTTP协议是由从客户机到服务器的请求(Request)和从服务器到客户机的响应(Response)进行了约束和规范。

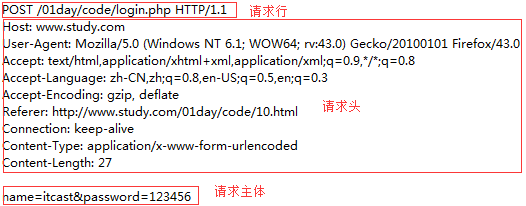
即HTTP协议主要由请求和响应构成。



常用请求方法 POST、GET、PUT、DELETE

### 请求/请求报文

请求由客户端发起，其规范格式为：请求行、请求头、请求主体。



1、请求行



由请求方式、请求URL和协议版本构成

2、请求头

请求头包含了客户端（浏览器）的相关信息，可以将这些信息发送至服务端，以便服务端能够获得。

Host：localhost请求的主机

Cache-Control：max-age=0控制缓存

Accept：\*/\* 接受的文档MIME类型

User-Agent：很重要

Referer：从哪个URL跳转过来的

Accept-Encoding：可接受的压缩格式

3、请求主体

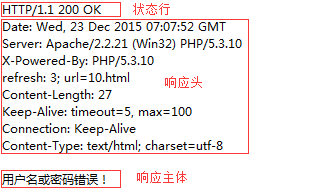
即传递给服务端的数据，格式为key=val&key1=val1，get方式请求时，请求主体为空，get方式请求的数据在请求url后面，格式为url?key=val&key1=val1。

**注：通过对比总结得到，当以post形式提交表单的时候，请求头里会设置**

**Content-Type: application/x-www-form-urlencoded，以get形式则不需要。**

### 响应/响应报文

响应由服务器发出，其规范格式为：状态行、响应头、响应主体。



1、状态行



由协议版本号、状态码和状态信息构成

2、响应头

Date：响应时间

Server：服务器信息

Content-Length：响应主体长度

Content-Type：响应资源的MIME类型

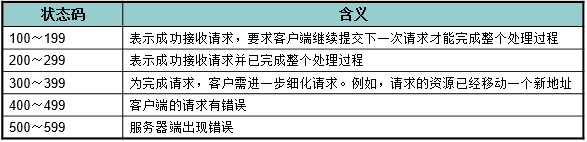
**MIME是标识文件类型的，文件后缀并不能正确无误的标识文件的类型。**

**客户端与服务器在进行数据传输的时候都是以字节形式进行的，咱们可以理解成是以“文本形式”传输，这时浏览器就需要明确知道该怎么样来解析这些文本形式的数据，MIME就是明确告知浏览器该如何来处理。**

3、响应主体

即服务端返回给客户端的内容；

状态码



常见的有200代表成功、304文档未修改、403没有权限、404未找到、500服务器错误

见代码示例login.html和login.php

### 调试工具

利用HTTP抓包工具在开发中可以帮我们进行调试，常用抓包工具HttpWatch、Fiddler、Charles、Firebug等

**浏览器插件**

Firebug、HttpWatch、chrome dev tools

**代理软件**

Charles、Fiddler

# AJAX编程

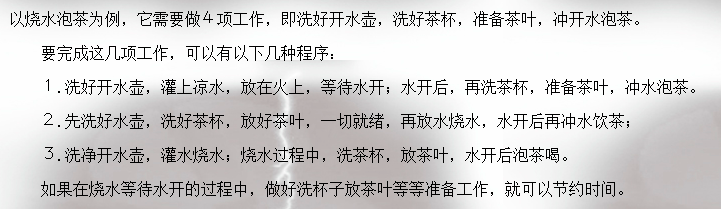
即 Asynchronous Javascript And XML，AJAX 不是一门的新的语言，而是对现有技术的综合利用。

本质是在HTTP协议的基础上以异步的方式与服务器进行通信。

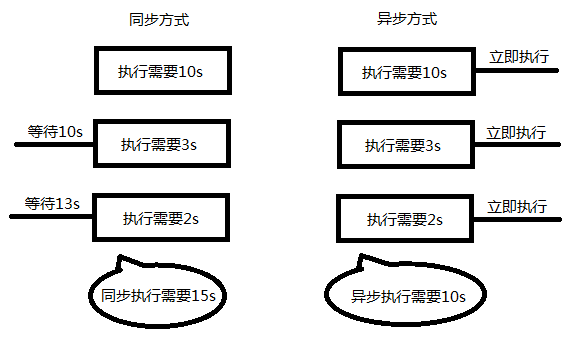
## 异步

指某段程序执行时不会阻塞其它程序执行，其表现形式为程序的执行顺序不依赖程序本身的书写顺序，相反则为同步。

其优势在于不阻塞程序的执行，从而提升整体执行效率。



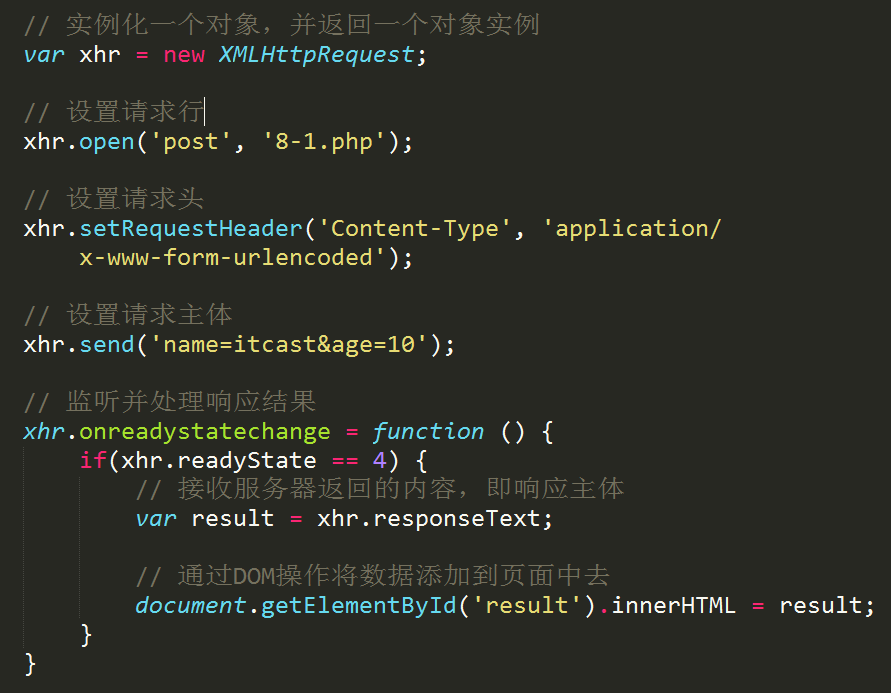
**对比同步异步程序执行**



## XMLHttpRequest

浏览器内建对象，用于在后台与服务器通信(交换数据) ，由此我们便可实现对网页的部分更新，而不是刷新整个页面。

下面是一个简单的例子



见代码示例8-1.html和8-1.php

由于XMLHttpRequest本质基于HTTP协议实现通信，所以结合HTTP协议和上面的例子我们分析得出如下结果：

### 请求

HTTP请求3个组成部分与XMLHttpRequest方法的对应关系：

1、请求行



2、请求头



**get请求可以不设置**

3、请求主体



**要按上述的1、2、3的顺序步骤书写**

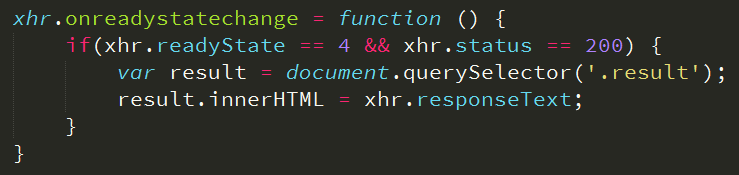
### 响应

HTTP响应是由服务端发出的，作为客户端更应关心的是响应的结果。

HTTP响应3个组成部分与XMLHttpRequest方法或属性的对应关系：

由于服务器做出响应需要时间（比如网速慢等原因），所以我们需要监听服务器响应的状态，然后才能进行处理。

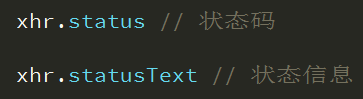
一次请求/响应的过程被划分成了5个阶段，分别对应0~4五个数字，xhr.readyState可以得到这个状态值，其中xhr.readyState == 4时，表示响应完成，这时我们处理响应结果才最有意义，xhr.status == 200，表示服务器的响应码，一个请求/响应始终包括0~4个状态，即使文件未找到（404），这时我们不可能会获得响应数据，为了保证程序逻辑的严谨性，我们需要判断xhr.status == 200时，然后再处理后续的数据处理逻辑。



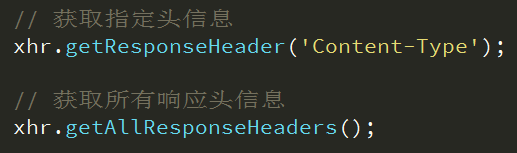
**onreadystatechange是Javascript的事件的一种，其意义在于监听XMLHttpRequest的状态。**

见代码示例8-1.html和8-1.php

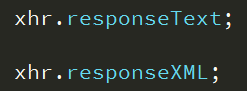
1、获取状态行（包括状态码&状态信息）



2、获取响应头



3、响应主体



我们需要检测并判断响应头的MIME类型后确定使用request.responseText或者request.responseXML

### API详解

xhr.open() 发起请求，可以是get、post方式

xhr.setRequestHeader() 设置请求头

xhr.send() 发送请求主体get方式使用xhr.send(null)

xhr.onreadystatechange = function () {} 监听响应状态

xhr.readyState = 0时，UNSENT open尚未调用

xhr.readyState = 1时，OPENED open已调用

xhr.readyState = 2时，HEADERS\_RECEIVED 接收到头信息

xhr.readyState = 3时，LOADING 接收到响应主体

xhr.readyState = 4时，DONE 响应完成

不用记忆状态，只需要了解有状态变化这个概念

xhr.status表示响应码，如200

xhr.statusText表示响应信息，如OK

xhr.getAllResponseHeaders() 获取全部响应头信息

xhr.getResponseHeader('key') 获取指定头信息

xhr.responseText、xhr.responseXML都表示响应主体

**注：GET和POST请求方式的差异（面试题）**

1、GET没有请求主体，使用xhr.send(null)

2、GET可以通过在请求URL上添加请求参数

3、POST可以通过xhr.send('name=itcast&age=10')

4、POST需要设置



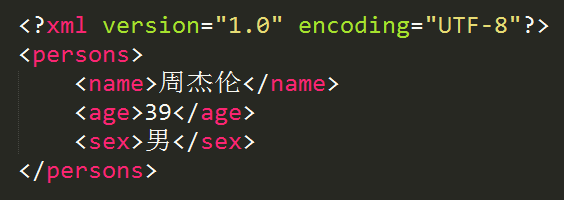
5、GET效率更好（应用多）

6、GET大小限制约4K，POST则没有限制

问题？如何获取复杂数据呢？

## XML

XML是一种标记语言，很类似HTML，其宗旨是用来传输数据，具有自我描述性（固定的格式的数据）。



### 语法规则

1、必须有一个根元素

2、不可有空格、不可以数字或.开头、大小写敏感

3、不可交叉嵌套

4、属性双引号（浏览器自动修正成双引号了）

5、特殊符号要使用实体

6、注释和HTML一样

见代码示例8-3.xml

### XML示例

见代码示例stars.html、stars.php、stars.xml

在实际工作中xml数据是由PHP连接数据库取出数据后生成的，我们没有数据库的知识，所以我们通过手工创建了xml文档（stars.xml）然后通过PHP读取这个文档，并返回到客户端。

虽然可以描述和传输复杂数据，但是其解析过于复杂并且体积较大，所以实现开发已经很少使用了。

## JSON

即 JavaScript Object Notation，另一种轻量级的文本数据交换格式，独立于语言。

见代码示例8-4.html和8-4.php

### 语法规则

1、数据在键/值对中

2、数据由逗号分隔(最后一个健/值对不能带逗号)

3、花括号保存对象，方括号保存数组

4、使用双引号

见代码示例8-1.json

### JSON解析

JSON数据在不同语言进行传输时，类型为字符串，不同的语言各自也都对应有解析方法，需要解析完成后才能读取

**1、Javascript 解析方法**

eavl()、JSON对象 JSON.parse()、JSON.stringify()

见代码示例

JSON兼容处理json2.js

eval("("+data+")");//转换为json对象

Session: 服务器中

Cookie: 浏览器 中 4K 获取数据不方便

Localstorage : 20M 永久存储 setItem(k,v) getItme(k);

Sessionstorage： 5M 浏览器窗口关闭 数据销毁

**2、PHP解析方法**

json\_encode()、json\_decode()

总结：JSON体积小、解析方便且高效，在实际开发成为首选。

见代码示例8-5.html和8-5.php

## 兼容性

IE5、IE6中使用 ActiveObject("Microsoft.XMLHTTP")

如下图



关于IE的兼容方面，了解即可。

## 封装AJAX工具函数

为了提升我们的开发效率，我们自已将XMLHttpRequest封装成一个函数。

见代码示例ajax.html和ajax.php

## jQuery中的Ajax

jQuery为我们提供了更强大的Ajax封装，提供了不同的方法来实现：

1、$.ajax() 可配置方式发起Ajax请求

见代码示例ajax.html和ajax.php

**配置参数详解：**

|  |  |
| --- | --- |
| **配置参数** | **含义** |
| **url** | 请求（接口）地址 |
| **type** | 请求方式 |
| **dataType** | 约束服务器数据返回格式 |
| **data** | 发送至服务器的数据 |
| **timeout** | 超时设置 |
| **beforeSend** | 请求发起前的回调 |
| **success** | 响应成功时的回调 |
| **error** | 响应失败时的回调 |
| **complete** | 响应完成（成功/失败）时的回调 |

2、$.get() 以GET方式发起Ajax请求

见代码示例get.html和get.php

3、$.post() 以POST方式发起Ajax请求

见代码示例post.html和post.php

4、$.getJSON() 以GET方式请求json数据

见代码示例getJSON.html和stars.json

5、$.getScript() 以GET方式请求javascript脚本

见代码示例getScript.html和getScript.js

6、$('selector').load('xxx.html') 以GET方式加载html片段

见代码示例load.html和template.html

**其它方法：**

1、$('form').serialize() 序列化表单（即格式化key=val&key=val）方便表单的处理。

见代码示例serilize.html

2、$.ajaxSetup({}) 全局配置Ajax请求

见代码示例ajaxSetup.html

jQuery在XMLHttpRequest内置对象的基础之上结合现实开发中的需求封装了很多方法，并且可以灵活配置参数实现不同的业务需求，然而有些回调方法XMLHttpRequest对象并没有，比如beforeSend和complete回调方法，这些方法是jQuery进行的封装，其目的就是为了提升开发效率。

## 案例练习

1、Loading状态

2、禁止重复提交

3、表单处理

4、数据验证

# 模板引擎

## 原理剖析

其本质是利用正则表达式，替换模板当中预先定义好的标签。

**正则表达式exec用法**

**http://www.w3school.com.cn/jsref/jsref\_exec\_regexp.asp**

**封装简易模板引擎**

**见代码示例9-1.html、9-2.html和9-3.html**

## 流行模板引擎

**BaiduTemplate**：http://tangram.baidu.com/BaiduTemplate/

**ArtTemplate：**https://github.com/aui/artTemplate

**velocity.js：**https://github.com/shepherdwind/velocity.js/

**Handlebars：**http://handlebarsjs.com/

**参考资料:**

http://blog.jobbole.com/56689/

## artTemplate

1、引入template-native.js

2、<% 与  %> 符号包裹起来的语句则为模板的逻辑表达式

3、<%= content %>为输出表达式

## 案例练习

1、Ajax分页（瀑布流加载）

# 同源&跨域

## 同源策略

同源策略是浏览器的一种安全策略，所谓同源是指，域名，协议，端口完全相同。

见代码示例10-1.html

## 跨域

不同源则跨域

例如http://www.example.com/

|  |  |  |
| --- | --- | --- |
| http://api.example.com/detail.html | 不同源 | 域名不同 |
| https//www.example.com/detail.html | 不同源 | 协议不同 |
| http://www.example.com:8080/detail.html | 不同源 | 端口不同 |
| http://api.example.com:8080/detail.html | 不同源 | 域名、端口不同 |
| https://api.example.com/detail.html | 不同源 | 协议、域名不同 |
| https://www.example.com:8080/detail.html | 不同源 | 端口、协议不同 |
| http://www.example.com/detail/index.html | 同源 | 只是目录不同 |

**跨域访问会受到限制：**

1、不允许Javascrip进行DOM操作

见代码示例10-2.html和10-3.html

2、不允许XMLHttpRequest（Ajax）请求

见代码示例10-4.html和http://api.study.com/10-1.php

## 跨域解决方案

同源策略保证了不同域网站相互访问的安全性，但是在实际开中跨域访问数据又是必不可少的，比如https://www.taobao.com和https://api.m.taobao.com虽然属于跨域，但是需要数据的相互访问，以下汇总了系列解决方案。

1、document.domain + iframe顶级域名相同的可以通过domain.name来解决，即同时设置 document.domain = 顶级域名（如example.com）

2、window.name + iframe

3、location.hash + iframe

4、window.postMessage()

**参考资料**

**http://rickgray.me/2015/09/03/solutions-to-cross-domain-in-browser.html**

以上几种解决方案多数只能在某一特定情形下使用，兼容性及通用性都不够好，更通用的是JSONP解决方案。

## JSONP

JSON with Padding

**1、原理剖析**

其本质是利用了<script src=""></script>标签具有可跨域的特性，由服务端返回一个预先定义好的Javascript函数的调用，并且将服务器数据以该函数参数的形式传递过来，此方法需要前后端配合完成。

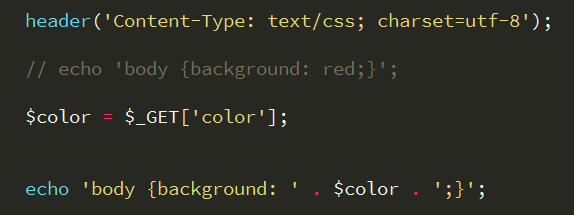
JOSNP 只能使用get请求方式  
 <script src="http://www.baidu.com/01.php?name=zs"></script>

见代码示例10-5.html

1. 文件后缀并不能代表什么，关建我们要看服务器返回的内容，比如css.php

这时返回的内容为 echo ‘body {background:color}’我们可以通过content-type:text/css来指定浏览器该如何解析，返回的内容

见代码示例10-6.html和 css.php



1. 同样我们也可以js文件以.php结尾，比如js.php 这时我们可以指定Content-Type: text/javascript可告知浏览器要以javascript来执行我们返回的内容，如果返回的是一个javascript的一个函数调用，那么在这个函数调用过程中可以将跨域请求来的数据以“实参”的传递过来，并且这个实参一般是JSON格式的

见代码示例10-7.html和js.php



1. 结合html标签src具有跨域访问的特性和Content-Type指定文档类型两个方面，可以实现跨域的数据访问。

见代码示例10-8.html和http://api.study.com/data.php

JSONP其本质就是服务端返回了一个Javascript的一个函数调用，而这个函数已经提前被定义了。

## jQuery中的JSONP

jQuery 的$.ajax() 方法当中集成了JSONP的实现，可以非常方便的实现跨域数据的访问。

见代码示例10-9.html和http://api.study.com/jsonp.php

$.ajax()在jQuery中是一个函数/方法，既可以用来发起一个XMLHttpRequest的请求，也可以用来进行跨域访问，当用来跨域访问时其实是通过Javascript创建了一个<script src=""></script>，并且将这个标签的src指向跨域访问的处理程序，然后再将这个<script></script>从DOM树中删除。

实现机制，以下是伪代码



**jQuery中跨域访问：**

dataType: 'jsonp' 设置dataType值为jsonp即开启跨域访问

jsonp 可以指定服务端接收的参数的“key”值，默认为callback

jsonpCallback 可以指定相应的回调函数，默认自动生成

**案例练习**

1、天气预报

# XMLHttpRequest 2.0

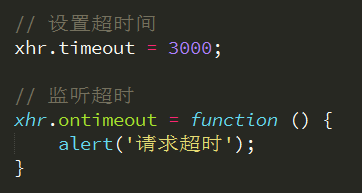
技术总是在实践中不断更新的，XMLHttpRequest也不例外。

## 设置超时

1. 设置超时 xhr.timeout
2. 监听超时事件 xhr.ontimeout = function () {// code}

当请求超时，此事件就会被触发

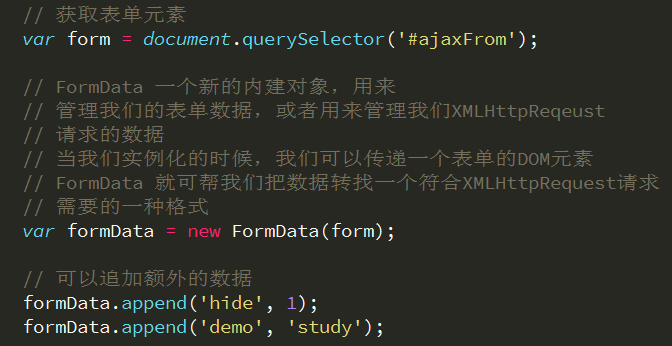
见代码示例11-1.html和11-1.php



## FormData

1. 提供了一个新的内建对象，可用于管理表单数据
2. 首先要获取一个表单元素form
3. 然后在实例化时 new FormData(form)，将表单元素form传进去
4. 会返回一个对象，此对象可以直接做为xhr.send(formData)的参数
5. 此时我们的数据就是以二进制形式传递了
6. 注意我们这里只能以post形式传递，浏览器会自动为我们设置一个合适的请求头

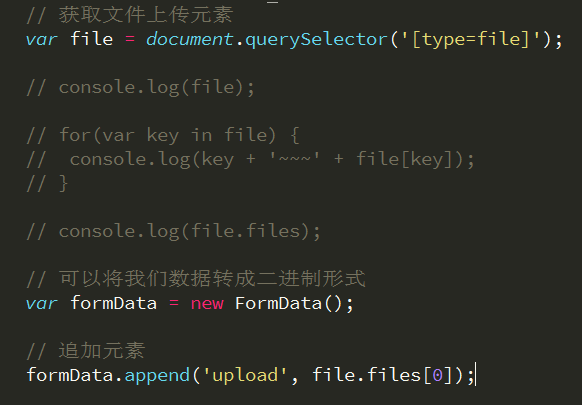
见代码示例11-2.html和11-2.php



## 二进制

1. 我们上传文件是以二进制形式传递的
2. 我们可以通过表单<input type=”file”>获取到一个文件对象
3. 然后file.files[0]可以获取文件信息
4. 然后再利用var formData = new FormData() 实例化
5. 然后再利用formData.append(‘upload’, file.files[0])将文件转成二进制
6. 最后将 formData 做为xhr.send(formData)的参数

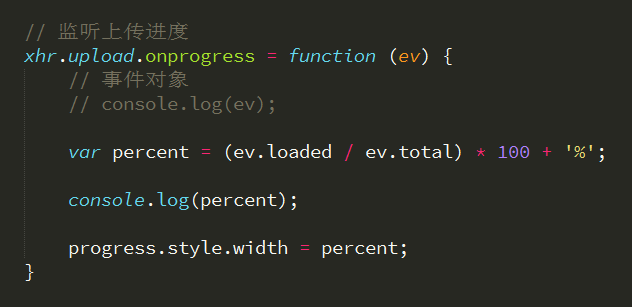
见代码示例11-3.html 和 11-3.php



## 上传进度

1. 利用XMLHttpRequest我们可以实现文件的上传
2. 并且我们可以通过xhr.upload.onprogress = function (ev) {// code}，监听上传的进度
3. 这时我们上传的进度信息会保存在事件对象ev里
4. ev.loaded 表示已上传的大小，ev.total表示文件整体的大小
5. var percent = ev.loaded / ev.total

见代码示例11-4.html和11-4.php



# 综合练习

## 服务端跨域

header( 'Access-Control-Allow-Origin:\*' );

header( 'Access-Control-Allow-Origin:http://www.study.com' );

## 接口化开发

请求地址即所谓的接口，通常我们所说的接口化开发，其实是指一个接口对应一个功能，并且严格约束了请求参数和响应结果的格式，这样前后端在开发过程中，可以减少不必要的讨论，从而并行开发，可以极大的提升开发效率，另外一个好处，当网站进行改版后，服务端接口只需要进行微调。

//接口文档

$.ajax({

Type: ?

Url: ?

Data: {id:123123}

Success:function(info){

}

});

天气接口：

<http://developer.baidu.com/map/carapi-7.htm>

百度API：

<http://apistore.baidu.com/>

密钥：

21f068f20fd5b7ca99c8908c1ae9f2bb

单页面应用

图灵机器人：http://www.tuling123.com/register/index.jhtml