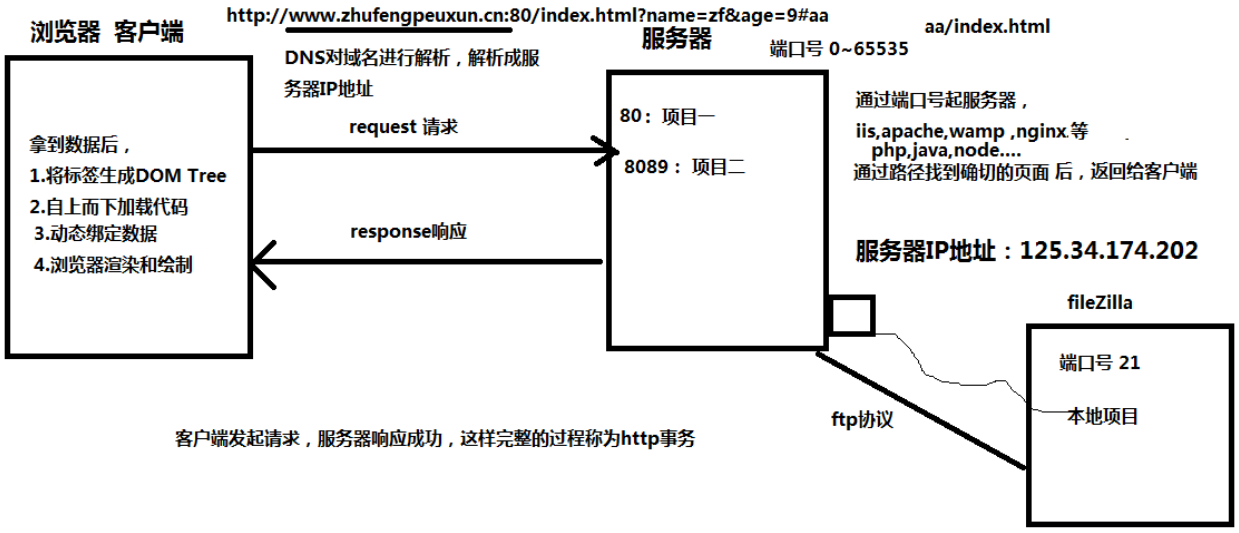
js正式课第七周



1.如何搭建博客？
购买服务器视频网址
http://www.html5train.com/kecheng/detail_894292?f=org_coursecenter

- 可以利用第三方平台
- 发布到github上（只能发布静态页面）
- 自己购买域名和服务器，然后进行页面优化和推广

2.利用购买域名和服务器的方式搭建博客

- 买（租）服务器 （阿里云买服务器）
  有了服务器后通过ftp（FileZilla）将本地项目传送到服务器上
- 购买域名 （万网）
  通过DNS数据库解析域名，将域名和服务器IP地址绑定
  www.zhufengpeixun.cn 125.39.174.202
- 开发项目，把项目发布到对应的端口号（0~65535）的服务器的内存里

1. 在浏览器的地址栏中输入url地址并按回车，这个过程发生了什么？

- 通过http协议 客户端向服务器端发起请求
- 查看url中的域名，将域名解析成服务器ip地址，找到对应的服务器
- 查看端口号，找到对应的服务，在这个服务中通过路径找到对应的页面
- 服务器将对应的内容通过response响应返回给客户端
- 客户端拿到数据后,按照如下步骤将数据在页面显示出来
  - 1.将所有标签生成DOM Tree
  - 2.代码自上而下执行，当遇到 script/link/img…会按照上面步骤重新向服务器发起请求
  - 3.进行动态绑定数据（ejs模板 <%=.. >,字符串拼接的方式）处理
  - 4.浏览器进行渲染和绘制

什么是http事务？客户端请求+服务器响应成功，这样完整过程称为一条http事务
客户端向服务器发起请求时会携带一些数据，服务器端响应时也会向客户端发送一些数据，这些数据详细的信息称为http报文

详细讲解url地址

URL:统一资源定位符
URI:统一资源标识符 （通常指的就是URL）
URN:统一资源名称 (用的很少)
URI = URL+URN

1.协议
http://www.zhufengpeixun.cn/course/index.html?name=zf&age=9#aa
http 超文本传输协议 默认端口号 80
https http+ssL 多了一个加密协议（通常网站涉及到支付或者用户隐私等） 默认端口号8080
ftp 文件传输协议 （将本地项目传输到服务器上） 默认端口号 21

2.域名
www.zhufengpeixun.cn 一级域名
online.zhufengpeixun.cn 二级域名
html5train.online.zhufengpeixun.cn 三级域名

3.端口号
相当于房间号 范围是0~65535

1. 路径 (端口号后面,问号前面)
   通过路径从服务器查找到对应的文件
   /course/index.html

5.问号传参
格式：key = value 每一个参数之间以&连接
?name=zf&age=9
1.把客户端数据通过传参方式发送给服务器端
2.让服务端返回指定参数的数据

6.hash值 (跟http事务无关,跟锚点定位或路由跳转有关)
js获取：window.location.hash

【http报文组成】

1. General 通用首部
   Request URL: http://www.zhufengpeixun.cn/ 请求的地址
   Request Method: GET 请求的方式
   Status Code: 200 OK http状态码
   Remote Address: 125.39.174.202:80 主机的ip地址和端口号
   2.Request Headers 请求头
   Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,/;q=0.8
   接收的数据格式
   Accept-Encoding: gzip, deflate 压缩的格式
   Accept-Language: zh-CN,zh;q=0.9 语言
   Cache-Control: no-cache 不缓存
   Connection: keep-alive 长链接
   Cookie: **cfduid=d11b7d024136e2b856679f99aaedeeedf1538033148;**
   root_domain_v=.zhufengpeixun.cn; _qddaz=QD.dgi18u.qwjmzi.jmk98609;
   pgv_pvi=3035234304; tencentSig=9047068672; _qdda=3-1.1;
   Hm_lvt_5ca1e1efc366a109d783a085499d59d9=1539656707,1539749036,1540029419,1540260952;
   Hm_lvt_24b7d5cc1b26f24f256b6869b069278e=1540260983;
   Hm_lvt_418b1c90fa35dc210dd5d2284d9f9f29=1539746411,1540029388,1540259896,1540262296;
   _qddab=3-qcrur5.jnl4ei2u; IESESSION=alive; pgv_si=s2276654080;

Hm_lpvt_418b1c90fa35dc210dd5d2284d9f9f29=1540265546; _qddamta_2852156370=3-0
记录一些信息，会发送到服务器端
Host: www.zhufengpeixun.cn 域名
Pragma: no-cache
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/68.0.3440.106 Safari/537.36 浏览器信息

3.Response Headers 响应头
Connection: keep-alive 长链接
Content-Encoding: gzip 压缩的格式
Content-Type: text/html 响应的数据格式
Date: Tue, 23 Oct 2018 03:41:30 GMT 服务器端响应式时间 格林尼治时间
Last-Modified: Wed, 19 Sep 2018 08:41:34 GMT 缓存有关
Server: yunjiasu-nginx 通过nginx启动web服务

4.请求主体和响应主体
Query String Parameters 请求体的内容
response下看到响应的内容

mime类型数据（对象类型的数据/子类型的数据）：
text/html html标签
text/plain 文本
application/json json格式的数据
application/x-www-form-urlencoded 表单元素的数据
multipart/form-data 上传多个文件
image/gif gif图片
image/jpg jpg图片
application/zip 压缩的格式
audio/x-wav 设置音频文件格式
video/mpeg 设置视频的文件格式

ajax(async javascript and xml)

在不刷新页面的前提下，局部更新数据
【请求的方式】
GET 向服务器获取响应的内容
HEAD 向服务器获取响应头的内容
DELETE 让服务器删除某些内容
POST 把发送的内容放在请求主体里发送给服务器 通过增加内容时用post方式
PUT 把发送的内容放在请求主体里发送给服务器,返回的状态只有202 通常修改内容用put方式
OPTIONS 是一种探测性请求，客户端和服务器端连接后，查看服务器端支持的请求方式有哪些，然后再发起支持请求方式的请求
【get系列】
get|head|delete
【post系列】
post|put
【get系列请求方式和post请求方式有什么不同】

- get系列数据是放在url地址后面发送给服务器，post系列是把数据放在请求体里发送给服务器
- 发送的数据大小有区别，get系列 chrome 8kb firefox 7kb ie 2kb post系列没有大小限制

- get系列会有缓存问题，post系列没有
- post系列请求方式比get系列请求方式安全写，get系列的数据由于放在url地址，有可能数据会被劫持

xhr.readyState 请求状态状态码
0 UNSENT 创建ajax对象,还未发送请求
1 OPENDED 设置请求的地址 执行open后再获取 发送请求
2 HEADERS_RECEIVED 返回响应头的内容
3 LOADING 响应主体的内容正在返回
4 DONE 响应成功，数据全部返回

xhr.status http状态码
2XX 响应成功
3XX

- 301 永久转移（例如：更改域名，当你还是访问老的域名时，永远转移到新域名地址）
- 302 临时转移
  例如：一般用在服务器超负荷运载，当用户访问量超过服务器所能承载状态时，后期访问的用户临时转移到其他服务器上 ,又例如：会把图片临时转移到其他服务器上，以减轻这个服务器的压力
- 307 临时重定向
  例如之前网站协议是http,后期更改成了https协议，当用户访问还是http协议时会临时重定向到https协议
  - 304
    客户端向服务器端发送请求时，服务端接收请求后，要判断下之前有没有返回过，若返回过，直接返回304，让客户端从缓存中获取数据。从缓存中获取数据比从服务器获取数据快很多
    从缓存中获取数据有可能获取的不是最新数据，若想每次获取的都是最新数据，在请求的地址后面加一个随机数,例如：
    ="">http://www.zhufengpeixun.cn/wechat/index.html?= Math.random();
    4XX 【客户端问题】
  - 400 参数错误
  - 401 权限不够
  - 404 地址错误，页面找不到
    5XX 【服务器端问题】
  - 500 未知的服务器错误
  - 503 服务器超负荷

xhr的属性和方法

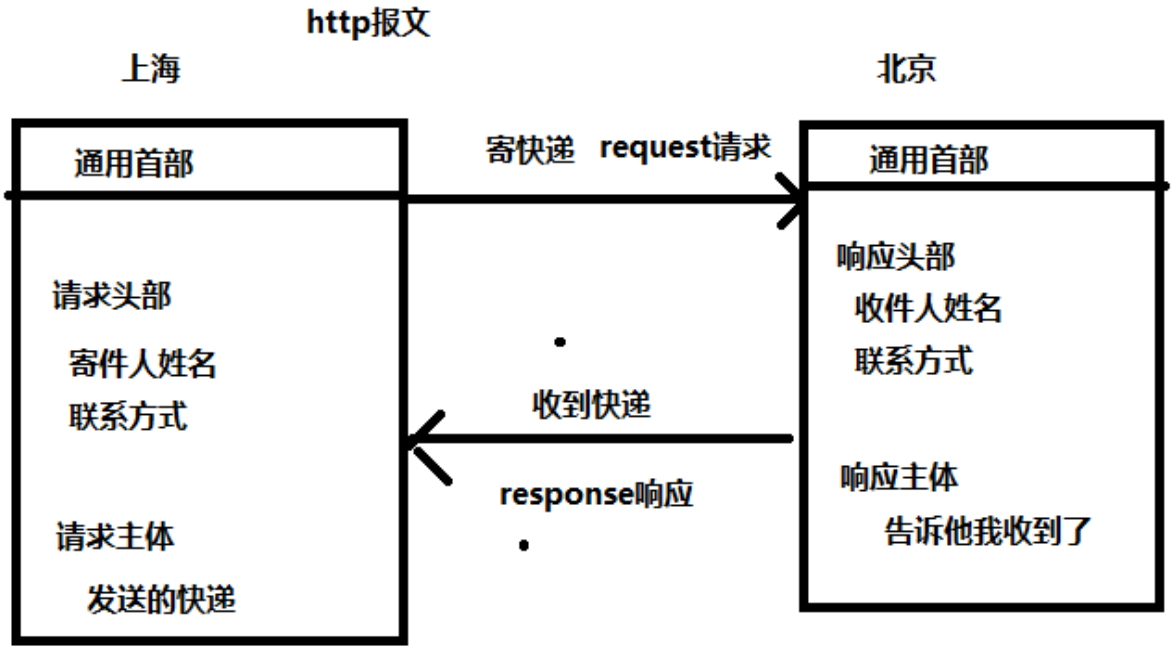
- xhr.onabort() 中止请求
- xhr.ontimeout = function(){} 设置超时后做些事情
- xhr.timeout = 5000 设置超时时间
- xhr.readyState 请求状态码
- xhr.response 目前各个浏览器还有兼容性问题
- xhr.responseText 获取响应的数据（string）
- xhr.responseXML xml格式的数据
- xhr.status http状态码
- xhr.statusText 状态码的描述信息
- xhr.withCredentials 是否允许跨域 true允许 false 不允许
- xhr.getAllResponseHeaders() 获取所有的响应头
- xhr.getResponseHeader() 获取指定的响应头
- xhr.overrideMimeType 重写mime类型

前后端交互调试bug

NetWork 下
All 所有请求的地址
XHR 通过ajax发送的请求，也就是所有的接口地址
JS 请求的js文件
css 请求的css文件
media 音视频
font 字体图标
职责问题 哪些属于前端问题，哪些属于后端问题

- 问号传参，传的参数错误 [前端问题]
- 请求体的数据提交错误（接口文档会指定key,而value就是客户端需要提交内容）【前端问题】
- 请求接口地址错误[前端问题]
- 服务端返回数据错误或者返回的数据跟接口文档指定的不一样[后端的问题]
- 返回状态码5XX [后端的问题]

http报文图



1.创建ajax对象

```
1.
var xhr = new XMLHttpRequest();
2.设置请求的链接地址
第一个参数表示请求的方式 get|head|post|put|options
第二个参数表示请求的接口地址 /add /delAA   data.json
第三个参数表示同步还是异步 默认true表示异步 false 表示同步
第四个和第五个参数设置user-name,user-password (一般不用)    xhr.open("get","data.jso
n",true);
//xhr.overrideMimeType('text/plain'); 不起作用
3.监听请求的状态    xhr.readyState 请求状态码 xhr.status  http状态码
xhr.onreadystatechange = function() {//事件 异步
```

```
        if (xhr.readyState === 4 && /^2\d{2}$/.test(xhr.status)) {
/*获取响应的数据*/
xhr.getResponseHeader("Date")//获取服务器响应时间
        }
};
4.将请求主体的内容发送给服务器
xhr.send(null) //send里放请求体的内容
```

jQ-ajax

```
$.ajax({
        url:"data.json",
        type:"get",/*请求的方式默认get方式*/
        async:true,/*默认为true*/
        cache:true,/*默认为true需要缓存，false不需要缓存*/
        contentType:"application/json",
        data:{},/*post设置请求体的数据 get将数据通过传参的方式发送的过去*/
        dataType:"json",/*xml,json,html,text,jsonp*/
        headers:{},/*设置请求头的信息*/
        beforeSend:function(){/*发送请求前执行*/

        },
        success:function(data){
            console.log(data);/*自动会转换成对象  响应数据*/
        },
        error:function(XMLHttpRequest, textStatus, errorThrown){ /*响应失败时执行的
回调函数*/
            console.log(errorThrown);
        },
        complete:function(){/*请求结束后，不管成功或失败都会调用*/
        }
    })
```