# 网络笔记

目录

[网络笔记 1](#_Toc14952208)

[Url资源定位的交互 2](#_Toc14952209)

[当你在浏览器里输入一个url发生了什么？ 2](#_Toc14952210)

[三次握手 2](#_Toc14952211)

[四次挥手 3](#_Toc14952212)

[五层网络模型 4](#_Toc14952213)

[TCP/UDP（传输层协议） 4](#_Toc14952214)

[面向连接的TCP 4](#_Toc14952215)

[面向非连接的UDP协议 5](#_Toc14952216)

[应用层协议: http https等 5](#_Toc14952217)

[浏览器缓存机制（http） 7](#_Toc14952218)

[请求方法 GET POST的区别 8](#_Toc14952219)

[Ajax用途 9](#_Toc14952220)

[什么是ajax? 9](#_Toc14952221)

[Ajax对象方法 10](#_Toc14952222)

[表单提交数据时所用到的属性。 10](#_Toc14952223)

[Method 10](#_Toc14952224)

[action 10](#_Toc14952225)

[enctype 10](#_Toc14952226)

[同源策略 11](#_Toc14952227)

[www拓展 11](#_Toc14952228)

[解决跨域问题的几种办法 12](#_Toc14952229)

[JSONP原理 12](#_Toc14952230)

[什么是cookie？ 12](#_Toc14952231)

[为什么会有cookie？ 13](#_Toc14952232)

[Cookie内存大小受限制 13](#_Toc14952233)

[Cookie的特点 13](#_Toc14952234)

[Cookie与Session的区别 13](#_Toc14952235)

[Iframe 14](#_Toc14952236)

[iframe利弊 14](#_Toc14952237)

[如何获取iframe内的window 14](#_Toc14952238)

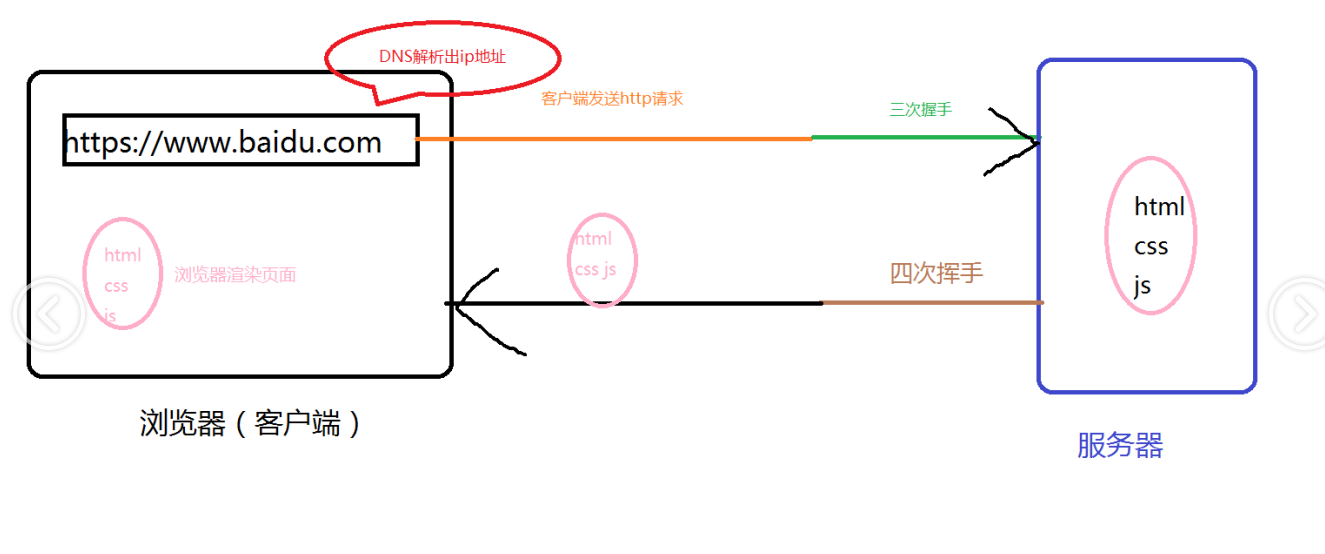
[父子页面窗口的关系 14](#_Toc14952239)

[父子窗口通信 15](#_Toc14952240)

[判断iframe加载完成 15](#_Toc14952241)

[iframe受跨域限制如何解决 15](#_Toc14952242)

## Url资源定位的交互



### 当你在浏览器里输入一个url发生了什么？

1.浏览器通过DNS域名解析到服务IP（ping www.baidu.com）

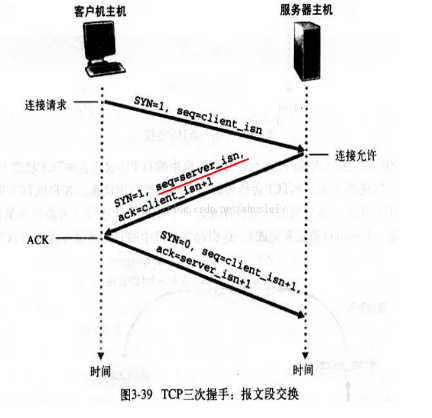
2.客户端(浏览器)通过TCP协议建立到服务器的TCP连接 (三次握手)

三次握手：

1.先Client端发送连接、请求报文。

2.Server端接受连接后回复ACK报文，并为这次连接分配资源。

3.Client端接收到ACK报文后也向Server端发送ACK报文，并分配资源，这样TCP连接就建立了。



3.客户端（浏览器）向web服务器端（HTTP服务器）发送HTTP协议包，请求服务器里的资源文档 （telnet 模拟）

4.服务器向客户端发送HTTP协议应答包

5.客户端和服务器断开（四次挥手），客户端开始解释处理HTML文档

四次挥手:

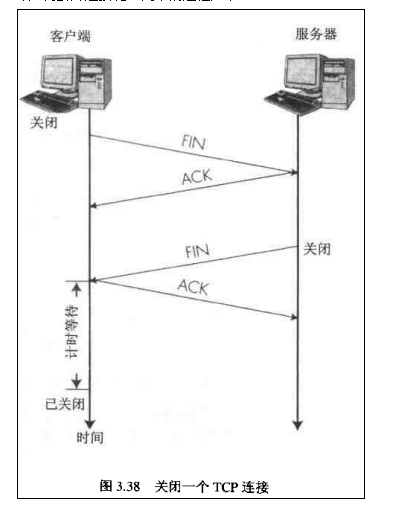
1.Client端发起中断连接请求，也就是发送FIN报文。Server端接到FIN报文后，意思是说"我Client端没有数据要发给你了"，但是如果你还有数据没有发送完成，则不必急着关闭（Socket），可以继续发送数据。

2.server发送ACK，"告诉Client端，你的请求我收到了，但是我还没准备好，请继续等我的消息"。

wait:这个时候Client端就进入FIN\_WAIT状态，继续等待Server端的FIN报文。

3.当Server端确定数据已发送完成，则向Client端发送FIN报文，"告诉Client端，好了，我这边数据发完了，准备好关闭连接了"。

4.Client端收到FIN报文后，"就知道可以关闭连接了，但是他还是不相信网络，怕Server端不知道要关闭，所以发送ACK后进入TIME\_WAIT状态，如果Server端没有收到ACK则可以重传。“，Server端收到ACK后，"就知道可以断开连接了"。Client端等待了2MSL后依然没有收到回复，则证明Server端已正常关闭，那好，我Client端也可以关闭连接了。Ok，TCP连接就这样关闭了！



五层网络模型：

应用层：http协议、DNS协议

运输层：TCP协议、UDP协议

网络层：IP地址（IP协议）

数据链路层：mac地址

物理层

### TCP/UDP（传输层协议）

#### 面向连接的TCP

TCP（Transmission Control Protocol，[传输控制协议](http://baike.baidu.com/view/544903.htm)）是基于连接的协议，也就是说，在正式收发数据前，必须和对方建立可靠的连接。一个TCP连接必须要经过三次“对话”才能建立起来，其中的过程非常复杂，我们这里只做简单、形象的介绍，你只要做到能够理解这个过程即可。

#### 面向非连接的UDP协议

“面向非连接”就是在正式通信前不必与对方先建立连接，不管对方状态就直接发送。与手机短信非常相似：你在发短信的时候，只需要输入对方手机号就OK了。

UDP（User Data Protocol，[用户数据报协议](http://baike.baidu.com/view/468464.htm)）是与TCP相对应的协议。它是[面向非连接](http://baike.baidu.com/view/2077319.htm)的协议，它不与对方建立连接，而是直接就把[数据包](http://baike.baidu.com/view/25880.htm)发送过去！

### 应用层协议: http https等

[超文本传输协议](http://baike.baidu.com/view/468465.htm)（HTTP，HyperText Transfer Protocol)是[互联网](http://baike.baidu.com/view/6825.htm)上应用最为广泛的一种[网络协议](http://baike.baidu.com/view/16603.htm)。

HTTPS（全称：Hyper Text Transfer Protocol over Secure Socket Layer），是以安全为目标的[HTTP](http://baike.baidu.com/view/9472.htm)通道，简单讲是HTTP的安全版。

上面的协议为了建立客户端与服务器端的连接，此协议为了让两者进行沟通。

为什么要有此协议呢，让计算机之间按照规矩说话，你问我答，你怎么问我怎么答，否则计算机各说各话，没办法沟通。

http （请求报文，响应报文） 通过报文进行沟通

请求报文：

请求头(Request): 请求方法（GET POST DELETE HEAD TRACE OPTION）

请求资源 (URL)

请求协议版本（HTTP/1.1）

请求行: <http://tools.jb51.net/table/http_header>

请求主体: 表单提交数据如：name=aimee&age=18;



响应报文：

响应头(Response): 响应协议版本号（HTTP/1.1）

响应状态码 （200）

响应状态文字 （0K）

<http://tool.oschina.net/commons?type=5>(响应状态码）

响应行: <http://tools.jb51.net/table/http_header>

响应主体: ‘sign success’ (注册成功)



响应状态码:



常见的http状态码

成功状态码：

200 服务器成功返回内容

301/2 永久/临时重定向

304 资源未被修改过

失败状态码：

404 请求内容不存在

500 服务器暂时不可用

503 服务器内部错误

### 浏览器缓存机制（http）

Date： 服务器响应内容日期

Cache-control：内容缓存时间

no-cache 不被缓存的，只不过每次在向客户端（浏览器）提供响应数据时，缓存都要向服务器评估缓存响应的有效性。

no-store 用于防止重要的信息被无意的发布。在请求消息中发送将使得请求和响应消息都不使用缓存。 根据缓存超时

max-age 指示客户机可以接收生存期不大于指定时间（以秒为单位）的响应。

min-fresh 指示客户机可以接收响应时间小于当前时间加上指定时间的响应。

max-stale 指示客户机可以接收超出超时期间的响应消息。如果指定max-stale消息的值，那么客户机可以 接收超出超时期指定值之内的响应消息。

Expires：内容保质期，表示存在时间，允许客户端在这个时间之前不去检查（发请求），等同max-age的效果。但是如果同时存在，则被cache-control的max-age覆盖。

网站如何统计用户从何点击而来?referer：如果从浏览器地址栏里直接输入地址请求头没有referer

304 上次缓存的资源没有改变------浏览器如何知道是否直接取缓存的内容？

理解部分 请求头 响应头

请求头：  
if-None-Match: 匹配etag  如果它修改了 不取缓存

If-Modified-Since：将先前服务器端发过来的最后修改时间戳发送回去

响应头：

etag --->标记图片资源

last-Modified (服务器最后修改的时间)和 etag 配合使用

ETags和If-None-Match的工作原理是在HTTP Response中添加ETags信息。当客户端再次请求该资源时，将在HTTP Request中加入If-None-Match信息（ETags的值）。如果服务器验证资源的ETags没有改变（该资源没有改变），将返回一个304状态；否则，服务器将返回200状态，并返回该资源和新的ETags。

### 请求方法 GET POST的区别

在http协议里实际上没有区别！

常规理解：

1.GET 使用URL 或Cookie 传参，而POST将数据，放在BODY中。 ? NAME = ‘CST’&AGE=18

2.GET 的URL会有长度上的限制， POST可以传输很多数据。

3.POST比GET安全。

但其实HTTP协议里没有规定POST数据就要放在BODY里， 也没有要求GET数据就一定要放在URL中而不能放在BODY中。

HTTP协议对GET和POST 都没有对数据的长度进行限制，两方面原因造成数据限制的原因

1.早期浏览器会对URL长度进行限制（浏览器URL输入框）

2.浏览器会对Content-length进行限制，这是为了服务器安全和稳定。

1.是基于什么前提的?如果什么前提都没有，不使用任何规范，只考虑语法和理论上的HTTP协议。

Get和Post几乎没有什么区别，只有名字不一样。

2.如果是基于RFC规范的。

(1).理论上的（Specification）：GET和POST具有相同语法，但是有不同的语义。Get是用来获取数据的，POST是用来发送数据的，其他方面没有区别。

(2).实现上的（Implementation）：各种浏览器，就是这个规范的实现者。

常见的不同：

1. GET数据在URL是可见的；POST请求不显示在URL中。
2. GET对长度是有限制的，POST长度是无限的。
3. GET请求的数据可以收藏为书签，post请求的数据不可收藏为书签。
4. GET请求后，按后退、刷新按钮无影响，post数据会被重新提交。
5. GET请求编码类型：application/x-www-form-url,

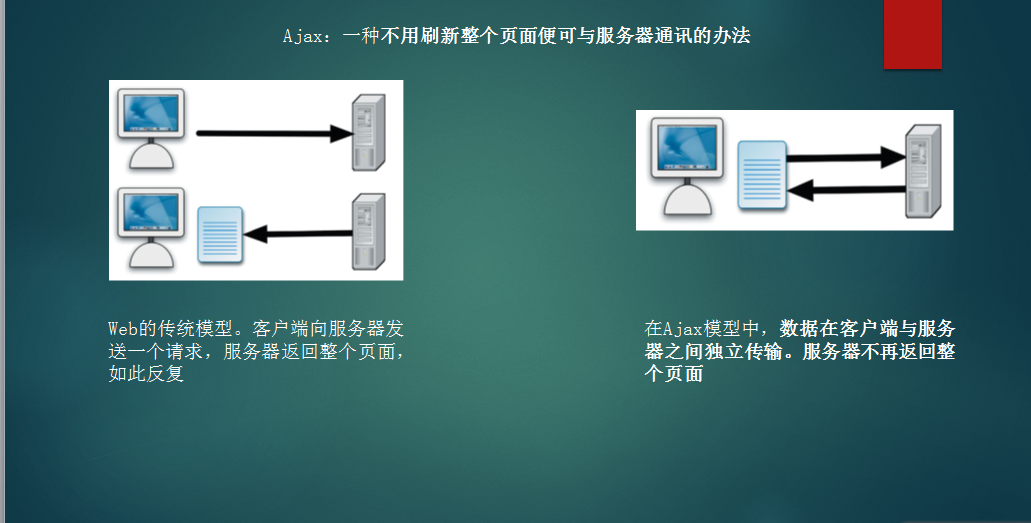
POST请求编码类型：有很多种。

Encodeapplication/x-www-form-urlencode

Multipart/form-data

1. GET历史参数会被保留在浏览器里，post不会保留在浏览器中。
2. GET只允许ASCⅡ编码格式，post没有编码限制，允许发二进制。
3. GET与POST相比，GET安全性较差，因为所发的数据是URL的一部分

## Ajax用途



## 什么是ajax?

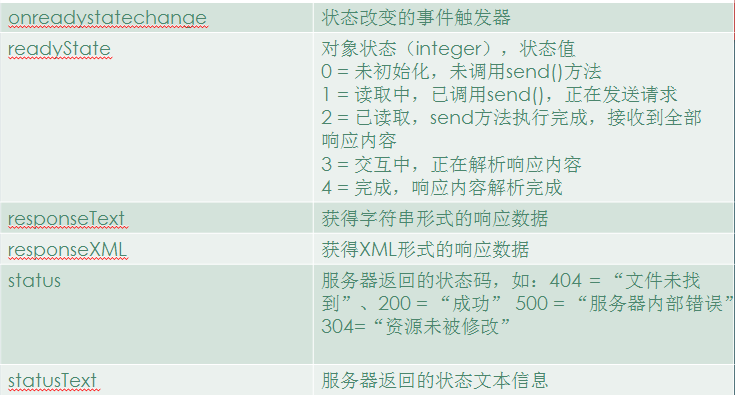
用JavaScript 以异步的形式操作 xml （现在操作的是json）（Asynchronous javascript and xml）

Ajax定义方法: new XMLHttpRequest()(除ie)

new ActiveXObject('Microsoft.XMLHttp')(ie定义ajax对象.)

## Ajax对象方法





## 表单提交数据时所用到的属性。

Method：GET POST等

action：address

enctype: 规定在发送表单数据之前如何对其进行编码

application/x-www-form-urlencoded在发送前编码所有字符（默认） multipart/form-data(<input type='file'>)不对字符编码。在使用包含文件上传控件的表单时，必须使用该值。

## 同源策略

浏览器有一个很重要的概念——同源策略(Same-Origin Policy)。所谓同源是指，域名，协议，端口相同。不同源的客户端脚本(javascript、ActionScript)在没明确授权的情况下，不能读写对方的资源。

简单的来说，浏览器不允许包含在腾讯页面的脚本访问阿里巴巴页面的数据资源，会受到同源策略的限制

协议:http或https;https是在http下加入了一个ssl层，是http的一个安全的版本;是加密了一些信息。

端口:http默认端口是80默认可以不写;https默认端口是443.

这里的同源指的是：同协议，同域名和同端口

域名是倒着解析的

.com 顶级域名分为两类:一类是按国家分的顶级域名;一类是按国际来分的顶级域名。

baidu.com （一）二级域名

zhidao.baidu.com （二）三级域名

www 二级域名前缀 表示万维网维护的

www.baidu.com 属于特殊的三级域名

zhidao.baidu.com 属于 百度自己维护的网络地址

www拓展:最开始，Internet提供的主要服务有万维网（WWW）、文件传输（FTP）、电子邮件（E-mail）、远程登录（Telnet）等。也就是说，那个时候的www（World Wide Web）是标识这是一个需要你用浏览器来访问的网页服务，而不是需要你用telnet访问的bbs，或者ftp工具访问的文件传输服务。所以那个时候，网站主页的域名前面要用www。

网址前面不加三w可以被叫做一个裸域名；1.裸域名是只能绑定dns的a记录的，不能绑定cname记录;a记录是指域名对应的ip地址;c记录是别名。2.裸域名的cookie作用域范围大。

baidu有多个服务区为他服务。不同的子域名对应处理不同服务的服务器现在，把任务分配到多台服务器，不需要通过子域名来区分了。http://www.goole.com一个地址背后有多台服务器支持运作。还猜用www只是尊重用户习惯，方便用户看。国外其实已经不用写www了。

com org net 属于顶级域名，是在全世界范围内解析的，cn hk 是在一个地区解析的，

如cn 中国

.com （商业机构）；

.net（从事互联网服务的机构）；

.org （非赢利性组织）；

.com.cn （国内商业机构）；

.net.cn （国内互联网机构）；

.org.cn （国内非赢利性组织）；

dns 先根据顶级域名判断网络范围在根据域名查找主机ip地址

前缀就不管了，理论上www开头相当于占位用的在国外一般不写www 国内风气就是写www。

Ajax 受同源策略的限制

## 解决跨域问题的几种办法

1.Flash （不做讨论）

2.服务器代理中转

3.Jsonp

4.document.domain(针对基础域名相同的情况)

bj.58.com document.domain = '58.com'

tj.58.com document.domain = '58.com'

## JSONP原理

1.Web页面上用<script> 引入 js文件时则不受是否跨域的影响（不仅如此，我们还发现凡是拥有"src"这个属性的标签都拥有跨域的能力，比如<script>、<img>、<iframe>）

2.于是我们把数据放到服务器上，并且数据为json形式（因为js可以轻松处理json数据）

3.因为我们无法监控通过<script>的src属性是否把数据获取完成，所以我们需要做一个处理。

4.实现定义好处理跨域获取数据的函数，如 function doJSON（data）{}。

5.用src获取数据的时候添加一个参数cb=‘doJSON’ (服务端会根据参数cb的值返回 对应的内容) 此内容为以cb对应的值doJSON为函数真实要传递的数据为函数的参数的一串字符 如 doJSON（’数据’）

Script标签，有src属性，所以可以发出网络请求；虽然可以引用其他域的资源，浏览器不限制，但是浏览器会将返回的内容，作为js代码执行

1. 判断页面请求与当页面的域是否同源，如果同源发送正常的ajax，就没有跨域的事情了。
2. 如果不同源生成一个script标签
3. 生成一个随机的callback名字，还得创建一个名为这个的方法
4. 设置script标签的src，设置为要请求的接口。
5. 将callback作为参数拼接在后面
6. 后端接收到请求后，开始准备要返回的数据
7. 后端拼接数据，将要返回的数据用callback的值和括号包裹起来。
8. 浏览器接收到内容，会当做js代码来执行。

Jquery会先判断是否同源，如果同源，那么设置get就是get;设置post就是post；如果不同源无论设置什么，都改为get.

## 什么是cookie？

Cookie是由[服务器端](http://baike.baidu.com/view/7072073.htm)生成，发送给User-Agent（一般是浏览器），（服务器告诉浏览器设置一下cookie），浏览器会将Cookie以key/value保存到某个目录下的[文本文件](http://baike.baidu.com/view/40622.htm)内，下次请求同一网站时就发送该Cookie给[服务器](http://baike.baidu.com/view/899.htm)（前提是浏览器设置为启用cookie）。

Cookie就是一个小型文件（浏览器对cookie的内存大小是有限制的-------用来记录一些信息）

## 为什么会有cookie？

Web应用程序是使用HTTP协议传输数据的。

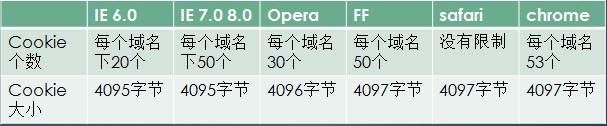
HTTP协议是无状态的协议。

一旦数据交换完毕，客户端与服务器端的连接就会关闭，再次交换数据需要建立新的连接。这就意味着服务器无法从连接上跟踪会话。

Cookie 具有保质期,分为会话cookie(临时cookie)和永久cookie(可以设置保存时间).

## Cookie内存大小受限制

Cookie有个数和大小的限制，大小一般是4k



## Cookie的特点

满足同源策略：虽然网站images.google.com与网站www.google.com同属于Google，但是域名不一样，二者同样不能互相操作彼此的Cookie。

问题来了举个例子：访问zhidao.baidu.com 再访问wenku.baidu.com还需要重新登陆百度账号吗？

设置document.domain = ‘baidu.com’;让页面属于这个基础域名下（那么此页面和任何二级域名为baidu.com的）设置限制----页面确实属于该基础域名之下那么资源公用了（ajax） cookie 共用了

Cookie 在本地可以被更改文件,敏感的数据不要放在cookie里

Cookie实际上主要是web服务器开发人员设置的，前端开发人员较少使用cookie。

Cookie是有限制的。

Cookie是存放在浏览器里的，不是存在某个页面上的。是可以长期储存的。

Cookie即使是保存在浏览器里，也是存在不同的域名下的。

## Cookie与Session的区别

Cookie存在浏览器上，Session存在服务器上。

Cookie是可以被别人复制走的，Session是很难被别人复制走的。

当用户量非常大时，是不容易存在Session中的，服务器端很耗费资源的。

后端可能不止一台服务器，用户的登录信息，一般只存在一台服务器上。因为用户的登录操作，在哪台机器上执行的，就一般存在哪台机器上。

## Iframe

Iframe就是一个标签dom元素，可以一个网页里嵌入另一个网页；导航栏tab切换页（古老的做法）、在线编辑器、广告植入；历史记录管理，解决ajax化网站响应浏览器前进后退按钮的方案；跨域通信等

### iframe利弊

触发 window 的 onload 事件是非常重要的。onload 事件触发使浏览器的 “忙” 指示器停止，告诉用户当前网页已经加载完毕。当 onload 事件加载延迟后，它给用户的感觉就是这个网页非常慢。

　　window 的 onload 事件需要在所有 iframe 加载完毕后(包含里面的元素)才会触发。通过 JavaScript 动态设置 iframe 的 SRC 可以避免这种阻塞情况

解决跨域问题 这个很牵强... 其实他已经不怎么用了

### 如何获取iframe内的window

获取子窗口

1.document.getElementsByTagName('iframe')[0].contentWindow

2.document.getElementsById('id').contentWindow

简易写法

window.frames['iframe的name']

IE专用

3.document.iframes[name].contentWindow

4.document.iframes[i].contentWindow

### 父子页面窗口的关系

window.self：就是自己--->神经病的属性

window.parent：父级窗口对象

window.top：顶级窗口对象

### 父子窗口通信

父访问子 (等待子iframe加载完成后可以通过iframe.contentWindow.变量 访问也会涉及跨域问题)

子访问父涉及跨域问题

### 判断iframe加载完成

非ie下使用onload事件：iframe(dom元素).onload = function () {}

ie下使用onreadystatechange或者设定计时器

iframe.onreadystatechange = function(){

if (iframe.readyState == "complete" || iframe.readyState == ''loaded”){

alert("Local iframe is now loaded.");

}

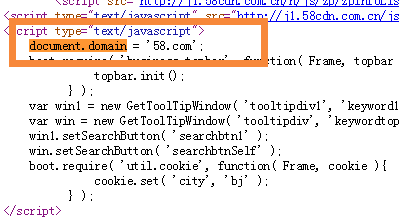
}

### iframe受跨域限制如何解决

1. document.domain: 解决跨域限制最好的办法

document.domain（bj.58.com;tj.58.com）

两个域名必须属于同一个基础域名!而且所用的协议，端口都要一致，否则无法跨域。



1. window.location.hash: 解决子页面访问父页面数据问题(window.location.href)

通过锚点#传递数据。利用定时器实时监测

锚点特性#不刷新页面

定时器耗性能方法比较笨！

子访问父

大概思路child.html 设置定时器 监测 url#query

parent.html (设置iframe dom元素 的src属性)url#query 设置参数

父访问子？

location.hash缺点

1、传递数据量有限

2、不太安全

3.window.name: 解决父页面访问子页面的数据问题

有三个页面：

window.name 特点 页面重载刷新 name值不变 即使换了一个页面

a.com/app.html：应用页面。

a.com/proxy.html：代理文件，一般是一个没有任何内容的html文件，需要和应用页面在同一域下。

b.com/data.html：应用页面需要获取数据的页面，可称为数据页面。

4.H5 PostMessage ---> 等讲到h5时讲