**扩展：**

**页面布局垂直居中，三栏布局**（看课件）

**清浮动的方法：**

1. 、给父级加高度（适用于确定父级高度和子级数量的情况下）
2. 、给父级里的最后加一个空标签，并设置clear样式
3. 、给父级加clear样式（适用于父级也需要float情况下）
4. 、父级：after{

content:’’;

clear:both;

display:block;

}

父级：{ zoom :1; } //兼容ie时添加

1. 、overflow : hidden; // (BFC)

**BFC：**（Block Formating Context）块级格式化上下文

触发一个元素的BFC相当于在元素外面建立一堵围墙，让它里面的元素不和外面的元素发生交集。不管内部如何布局，都与这个区域外部毫不相干。

作用： 1>、清除浮动

1. 、不被浮动所覆盖
2. 、阻止父子margin传递
3. 、解决ie6下的bug

触发条件： 1>、float不为none

1. 、overflow不为visible
2. 、display的值为inline-block、table-caption和table-cell之一
3. 、position的值不为static、relative之一

应用场景：常用的就是解决边距重叠，清除浮动的问题

**相对定位、绝对定位和固定定位**

相对定位： position：relative

1. 、不影响元素本身特性
2. 、不使元素脱离文档流（偏移之前的位置仍被保留）
3. 、如果不给元素偏移量，对元素无影响
4. 、提升层级

偏移量是相对自身原来的位置移动

绝对定位： position：absolute

1. 、元素移动后脱离文档流
2. 、使内嵌支持宽高（即内联元素）
3. 、块元素内容撑开宽度
4. 、提升层级
5. 、有定位父级（relative）时相对父级发生偏移，没有定位父级时是相对页面左上角 发生偏移
6. 、一般和相对定位结合使用

偏移量是相对定位父级的位置移动。

固定定位： position：fixed

与绝对定位的特性基本一致

差别在于 固定定位是相对于可视区进行定位

滚动条滚动后内容仍固定在其一开始的位置

**兼容性**

1. ie6、7下，不支持块标签的display：inline-block；

解决方法：

在 display：inline-block 下加：

\*display:inine;

\*zoom:1;

1. 在ie6下，块属性元素的高度小于19px时，高度会按19px处理

解决方法：overflow：hidden；

1. 在谷歌下，文字大小小于12px时，会按12px处理

解决方法：用图片代替

1. 在ie6下，1px的点线会显示成虚线

解决方法：用图片代替

1. 在ie6下给边框加transparent时，border-style最好设置成dashed或dotted， 才能实现transparent效果
2. ie6不兼容html5标签

解决方法：通过js动态创建html5标签（看js课件）

1. csshack：浏览器厂商为解决一些问题而留的后门，一般针对ie

|  |  |  |
| --- | --- | --- |
| 加  在  样式前 | - | IE6 |
| \* | IE6、7 |
| + | IE7及以下 |
| \0 | IE11及以下，除去IE5，IE7。IE6不兼容（不见了） |
| \9 | IE10及以下 |

如： +background：#666；

1. 条件注释：

<!--[ if IE 8 ]>内容<! [endif]--> // 只被IE 8 解析

1. 清浮动 ：after方法

.clear：after{

content：“”；

display：block；

clear：both；

}

.clear{ \*zoom： 1； } // IE6、7兼容

1. 在IE6下，父级有边框的话，可能会造成子元素的margin失效

解决方法：子元素触发haslayout

扩展：

haslayout： IE渲染引擎中的一个属性，用来调节元素的渲染模式，当我们把该属性设为true，这个属性就根据自身的内容大小或父级大小，来重新计算自己的宽高。

触发haslayout的样式：

display：inline-block;

height：（除auto的任何值）；

float：（left或right）；

position：absolute；

width：（除auto的任何值）；

zoom：（除normal的任何值）；

1. 同级有浮动元素，最好所有同级元素都浮动\*
2. Ie6下双边距bug

在ie6下，块元素有浮动和横向margin，横向的margin值会被放大成两倍（浮 动方向与margin方向一致时，该方向会出现双倍边距）

解决方法：块元素加display：inline

1. 浮动下margin失效

在ie6下，一行元素的宽（占位宽）度之和，与父级的宽度相差超过2px，则 最后一行元素的下margin失效

解决方法：

1. 、打破出现的规则（相差不超过2px）
2. 、子元素改成inline
3. 文字溢出：

Ie6下，子集元素的宽度和父级宽度相差小于3px，并且两个浮动子元素之间 有注释或内嵌元素，元素内容会被复制

解决方法：用div把注释或内嵌元素包起来

根本原因：宽度设置不严谨

要求同级元素：块跟块，或内跟内

1. li之间的间隙

Ie6、7下li本身没有加浮动，但是li的内容都浮动，li下边就会有4px的间隙

解决方法：

1. 、给li加浮动
2. 、给li加vertical-align：top；
3. 绝对定位元素消失

Ie6下，当浮动元素和绝对定位元素时同级关系，并且浮动元素的占位宽和父级相差小于3px，绝对定位元素会消失

解决方法：把绝对定位元素单独包起来

1. 定位偏移量误差

Ie6下，父级宽高是奇数的话，元素的right值和bottom值会有1px的偏差

解决方法：父级最好是偶数

1. 父级无法包含相对定位子级

Ie6、7下，子元素有相对定位时，父级的overflow包不住它

解决方法：给父级的父级也加相对定位

1. 输入类型控件间隙

Ie6、7下input上下各有1px的间隙

解决方法：给input加浮动

1. 输入类型控件背景图

Ie6、7下，输入类型的表单控件，输入文字时，背景图会随文字一块移动

解决方法：把背景图片给到父级

1. 输入类型控件边距

Ie6、7下，输入类型表单控件border：none；无效

解决方法： 1>、border：0；

2>、重置input的背景

22、ie6、7下，输入类型的表单控件，输入文字时，光标偏上

解决方法：给一个和高度一样的行高

1. png24图片在ie6浏览器上出现背景（即不正常状态）

解决方法：做成png8，或者引用js脚本处理（看js课件）

#### **如何解决跨域问题**

JSONP：

原理是：动态插入script标签，通过script标签引入一个js文件，这个js文件载入成功后会执行我们在url参数中指定的函数，并且会把我们需要的json数据作为参数传入。

由于同源策略的限制，XmlHttpRequest只允许请求当前源（域名、协议、端口）的资源，为了实现跨域请求，可以通过script标签实现跨域请求，然后在服务端输出JSON数据并执行回调函数，从而解决了跨域的数据请求。

优点是兼容性好，简单易用，支持浏览器与服务器双向通信。缺点是只支持GET请求。

JSONP：json+padding（内填充），顾名思义，就是把JSON填充到一个盒子里

<script>

function createJs(sUrl){

var oScript = document.createElement('script');

oScript.type = 'text/javascript';

oScript.src = sUrl;

document.getElementsByTagName('head')[0].appendChild(oScript);

}

createJs('jsonp.js');

box({

'name': 'test'

});

function box(json){

alert(json.name);

}

</script>

CORS

服务器端对于CORS的支持，主要就是通过设置Access-Control-Allow-Origin来进行的。如果浏览器检测到相应的设置，就可以允许Ajax进行跨域的访问。

通过修改document.domain来跨子域

将子域和主域的document.domain设为同一个主域.前提条件：这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，否则无法利用document.domain进行跨域

主域相同的使用document.domain

使用window.name来进行跨域

window对象有个name属性，该属性有个特征：即在一个窗口(window)的生命周期内,窗口载入的所有的页面都是共享一个window.name的，每个页面对window.name都有读写的权限，window.name是持久存在一个窗口载入过的所有页面中的

使用**[HTML5](http://lib.csdn.net/base/html5" \o "HTML5知识库" \t "https://blog.csdn.net/kebi007/article/details/_blank)**中新引进的window.postMessage方法来跨域传送数据

还有flash、在服务器上设置代理页面等跨域方式。个人认为window.name的方法既不复杂，也能兼容到几乎所有浏览器，这真是极好的一种跨域方法。

#### **XML和JSON的区别？**

(1).数据体积方面。

JSON相对于XML来讲，数据的体积小，传递的速度更快些。

(2).数据交互方面。

JSON与JavaScript的交互更加方便，更容易解析处理，更好的数据交互。

(3).数据描述方面。

JSON对数据的描述性比XML较差。

(4).传输速度方面。

JSON的速度要远远快于XML。

#### **说说TCP传输的三次握手四次挥手策略**

为了准确无误地把数据送达目标处，TCP协议采用了三次握手策略。用TCP协议把数据包送出去后，TCP不会对传送后的情况置之不理，它一定会向对方确认是否成功送达。握手过程中使用了TCP的标志：SYN和ACK。

发送端首先发送一个带SYN标志的数据包给对方。接收端收到后，回传一个带有SYN/ACK标志的数据包以示传达确认信息。   
最后，发送端再回传一个带ACK标志的数据包，代表“握手”结束。   
若在握手过程中某个阶段莫名中断，TCP协议会再次以相同的顺序发送相同的数据包。

断开一个TCP连接则需要“四次握手”：

第一次挥手：主动关闭方发送一个FIN，用来关闭主动方到被动关闭方的数据传送，也就是主动关闭方告诉被动关闭方：我已经不 会再给你发数据了(当然，在fin包之前发送出去的数据，如果没有收到对应的ack确认报文，主动关闭方依然会重发这些数据)，但是，此时主动关闭方还可以接受数据。

第二次挥手：被动关闭方收到FIN包后，发送一个ACK给对方，确认序号为收到序号+1（与SYN相同，一个FIN占用一个序号）。

第三次挥手：被动关闭方发送一个FIN，用来关闭被动关闭方到主动关闭方的数据传送，也就是告诉主动关闭方，我的数据也发送完了，不会再给你发数据了。

第四次挥手：主动关闭方收到FIN后，发送一个ACK给被动关闭方，确认序号为收到序号+1，至此，完成四次挥手。

#### **创建ajax的过程**

(1)创建`XMLHttpRequest`对象,也就是创建一个异步调用对象.

(2)创建一个新的`HTTP`请求,并指定该`HTTP`请求的方法、`URL`及验证信息.

(3)设置响应`HTTP`请求状态变化的函数.

(4)发送`HTTP`请求.

(5)获取异步调用返回的数据.

(6)使用JavaScript和DOM实现局部刷新.

var xmlHttp = new XMLHttpRequest();

xmlHttp.open('GET','demo.php','true');

xmlHttp.send()

xmlHttp.onreadystatechange = function(){

if(xmlHttp.readyState === 4 & xmlHttp.status === 200){

}

}

#### **HTTP和HTTPS**

HTTP协议通常承载于TCP协议之上，在HTTP和TCP之间添加一个安全协议层（SSL或TSL），这个时候，就成了我们常说的HTTPS。

默认HTTP的端口号为80，HTTPS的端口号为443。

#### **谈谈性能优化问题**

代码层面：避免使用css表达式，避免使用高级选择器，通配选择器。

缓存利用：缓存Ajax，使用CDN，使用外部js和css文件以便缓存，添加Expires头，服务端配置Etag，减少DNS查找等

请求数量：合并样式和脚本，使用css图片精灵，初始首屏之外的图片资源按需加载，静态资源延迟加载。

请求带宽：压缩文件，开启GZIP，

代码层面的优化

用hash-table来优化查找

少用全局变量

用innerHTML代替DOM操作，减少DOM操作次数，优化javascript性能

用setTimeout来避免页面失去响应

缓存DOM节点查找的结果

避免使用CSS Expression

避免全局查询

避免使用with(with会创建自己的作用域，会增加作用域链长度)

多个变量声明合并

避免图片和iFrame等的空Src。空Src会重新加载当前页面，影响速度和效率

#### **移动端性能优化**

* 尽量避免写在HTML标签中写Style属性
* 尽量使用css3动画，开启硬件加速。
* 适当使用touch事件代替click事件。
* 避免使用css3渐变阴影效果。
* 可以用transform: translateZ(0)来开启硬件加速。
* 不滥用Float。Float在渲染时计算量比较大，尽量减少使用
* 不滥用Web字体。Web字体需要下载，解析，重绘当前页面，尽量减少使用。
* 合理使用requestAnimationFrame动画代替setTimeout
* CSS中的属性（CSS3 transitions、CSS3 3D transforms、Opacity、Canvas、WebGL、Video）会触发GPU渲染，请合理使用。过渡使用会引发手机过耗电增加
* PC端的在移动端同样适用

相关阅读：[如何做到一秒渲染一个移动页面](https://github.com/cssmagic/blog/issues/20" \t "https://blog.csdn.net/kebi007/article/details/_blank)

#### **谈谈浮动和清除浮动**

浮动的框可以向左或向右移动，直到他的外边缘碰到包含框或另一个浮动框的边框为止。由于浮动框不在文档的普通流中，所以文档的普通流的块框表现得就像浮动框不存在一样。浮动的块框会漂浮在文档普通流的块框上。

#### **用过哪些设计模式？**

工厂模式：

主要好处就是可以消除对象间的耦合，通过使用工程方法而不是new关键字。将所有实例化的代码集中在一个位置防止代码重复。

工厂模式解决了重复实例化的问题 ，但还有一个问题,那就是识别问题，因为根本无法 搞清楚他们到底是哪个对象的实例。

function createObject(name,age,profession){//集中实例化的函数var obj = new Object();

obj.name = name;

obj.age = age;

obj.profession = profession;

obj.move = function () {

return this.name + ' at ' + this.age + ' engaged in ' + this.profession;

};

return obj;

}

var test1 = createObject('trigkit4',22,'programmer');//第一个实例var test2 = createObject('mike',25,'engineer');//第二个实例

构造函数模式

使用构造函数的方法 ，即解决了重复实例化的问题 ，又解决了对象识别的问题，该模式与工厂模式的不同之处在于：

1.构造函数方法没有显示的创建对象 (new Object());

2.直接将属性和方法赋值给 this 对象;

3.没有 renturn 语句。

#### **说说你对闭包的理解**

使用闭包主要是为了设计私有的方法和变量。闭包的优点是可以避免全局变量的污染，缺点是闭包会常驻内存，会增大内存使用量，使用不当很容易造成内存泄露。在js中，函数即闭包，只有函数才会产生作用域的概念

闭包有三个特性：

1.函数嵌套函数

2.函数内部可以引用外部的参数和变量

3.参数和变量不会被垃圾回收机制回收

#### **请你谈谈Cookie的弊端**

cookie虽然在持久保存客户端数据提供了方便，分担了服务器存储的负担，但还是有很多局限性的。

第一：每个特定的域名下最多生成20个cookie

1.IE6或更低版本最多20个cookie

2.IE7和之后的版本最后可以有50个cookie。

3.Firefox最多50个cookie

4.chrome和Safari没有做硬性限制

IE和Opera 会清理近期最少使用的cookie，Firefox会随机清理cookie。

cookie的最大大约为4096字节，为了兼容性，一般不能超过4095字节。

IE 提供了一种存储可以持久化用户数据，叫做userdata，从IE5.0就开始支持。每个数据最多128K，每个域名下最多1M。这个持久化数据放在缓存中，如果缓存没有清理，那么会一直存在。

优点：极高的扩展性和可用性

1.通过良好的编程，控制保存在cookie中的session对象的大小。

2.通过加密和安全传输技术（SSL），减少cookie被破解的可能性。

3.只在cookie中存放不敏感数据，即使被盗也不会有重大损失。

4.控制cookie的生命期，使之不会永远有效。偷盗者很可能拿到一个过期的cookie。

缺点：

1.`Cookie`数量和长度的限制。每个domain最多只能有20条cookie，每个cookie长度不能超过4KB，否则会被截掉.

2.安全性问题。如果cookie被人拦截了，那人就可以取得所有的session信息。即使加密也与事无补，因为拦截者并不需要知道cookie的意义，他只要原样转发cookie就可以达到目的了。

3.有些状态不可能保存在客户端。例如，为了防止重复提交表单，我们需要在服务器端保存一个计数器。如果我们把这个计数器保存在客户端，那么它起不到任何作用。

#### **浏览器本地存储**

在较高版本的浏览器中，js提供了sessionStorage和globalStorage。在HTML5中提供了localStorage来取代globalStorage。

html5中的Web Storage包括了两种存储方式：sessionStorage和localStorage。

sessionStorage用于本地存储一个会话（session）中的数据，这些数据只有在同一个会话中的页面才能访问并且当会话结束后数据也随之销毁。因此sessionStorage不是一种持久化的本地存储，仅仅是会话级别的存储。

而localStorage用于持久化的本地存储，除非主动删除数据，否则数据是永远不会过期的。

#### **cookie 和session 的区别：**

1、cookie数据存放在客户的浏览器上，session数据放在服务器上。

2、cookie不是很安全，别人可以分析存放在本地的COOKIE并进行COOKIE欺骗

考虑到安全应当使用session。

3、session会在一定时间内保存在服务器上。当访问增多，会比较占用你服务器的性能

考虑到减轻服务器性能方面，应当使用COOKIE。

4、单个cookie保存的数据不能超过4K，很多浏览器都限制一个站点最多保存20个cookie。

5、所以个人建议：

将登陆信息等重要信息存放为SESSION

其他信息如果需要保留，可以放在COOKIE中

#### **CSS中link 和@import的区别是？**

(1) link属于HTML标签，而@import是CSS提供的;

(2) 页面被加载的时，link会同时被加载，而@import被引用的CSS会等到引用它的CSS文件被加载完再加载;

(3) import只在IE5以上才能识别，而link是HTML标签，无兼容问题;

(4) link方式的样式的权重 高于@import的权重.

#### **说说你对语义化的理解？**

1，去掉或者丢失样式的时候能够让页面呈现出清晰的结构

2，有利于SEO：和搜索引擎建立良好沟通，有助于爬虫抓取更多的有效信息：爬虫依赖于标签来确定上下文和各个关键字的权重；

3，方便其他设备解析（如屏幕阅读器、盲人阅读器、移动设备）以意义的方式来渲染网页；

4，便于团队开发和维护，语义化更具可读性，是下一步吧网页的重要动向，遵循W3C标准的团队都遵循这个标准，可以减少差异化。

#### Doctype作用? 严格模式与混杂模式如何区分？它们有何意义?

1）、<!DOCTYPE> 声明位于文档中的最前面，处于 <html> 标签之前。告知浏览器以何种模式来渲染文档。

2）、严格模式的排版和 JS 运作模式是 以该浏览器支持的最高标准运行。

3）、在混杂模式中，页面以宽松的向后兼容的方式显示。模拟老式浏览器的行为以防止站点无法工作。

4）、DOCTYPE不存在或格式不正确会导致文档以混杂模式呈现。

#### **HTML与XHTML——二者有什么区别**

区别：

1.所有的标记都必须要有一个相应的结束标记

2.所有标签的元素和属性的名字都必须使用小写

3.所有的XML标记都必须合理嵌套

4.所有的属性必须用引号""括起来

5.把所有<和&特殊符号用编码表示

6.给所有属性赋一个值

7.不要在注释内容中使“--”

8.图片必须有说明文字

#### **常见兼容性问题？**

png24位的图片在iE6浏览器上出现背景，解决方案是做成PNG8.也可以引用一段脚本处理.

浏览器默认的margin和padding不同。解决方案是加一个全局的\*{margin:0;padding:0;}来统一。

IE6双边距bug:块属性标签float后，又有横行的margin情况下，在ie6显示margin比设置的大。

浮动ie产生的双倍距离（IE6双边距问题：在IE6下，如果对元素设置了浮动，同时又设置了margin-left或margin-right，margin值会加倍。）

#box{ float:left; width:10px; margin:0 0 0 100px;}

这种情况之下IE会产生20px的距离，解决方案是在float的标签样式控制中加入

\_display:inline;将其转化为行内属性。(\_这个符号只有ie6会识别)

渐进识别的方式，从总体中逐渐排除局部。

首先，巧妙的使用“\9”这一标记，将IE游览器从所有情况中分离出来。

接着，再次使用“+”将IE8和IE7、IE6分离开来，这样IE8已经独立识别。

css

.bb{

background-color:#f1ee18;/\*所有识别\*/

.background-color:#00deff\9; /\*IE6、7、8识别\*/

+background-color:#a200ff;/\*IE6、7识别\*/

\_background-color:#1e0bd1;/\*IE6识别\*/

}

怪异模式问题：漏写DTD声明，Firefox仍然会按照标准模式来解析网页，但在IE中会触发

怪异模式。为避免怪异模式给我们带来不必要的麻烦，最好养成书写DTD声明的好习惯。现在

可以使用[html5](http://www.w3.org/TR/html5/single-page.html)推荐的写法：`<doctype html>`

#### **DOM操作——怎样添加、移除、移动、复制、创建和查找节点。**

1）创建新节点

createDocumentFragment() //创建一个DOM片段

createElement() //创建一个具体的元素

createTextNode() //创建一个文本节点

2）添加、移除、替换、插入

appendChild()

removeChild()

replaceChild()

insertBefore() //并没有insertAfter()

3）查找

getElementsByTagName() //通过标签名称

getElementsByName() //通过元素的Name属性的值(IE容错能力较强，

会得到一个数组，其中包括id等于name值的)

getElementById() //通过元素Id，唯一性

#### **html5有哪些新特性、移除了那些元素？如何处理HTML5新标签的浏览器兼容问题？如何区分 HTML 和 HTML5？**

HTML5 现在已经不是 SGML 的子集，主要是关于图像，位置，存储，多任务等功能的增加。

拖拽释放(Drag and drop) API

语义化更好的内容标签（header,nav,footer,aside,article,section）

音频、视频API(audio,video)

画布(Canvas) API

地理(Geolocation) API

本地离线存储 localStorage 长期存储数据，浏览器关闭后数据不丢失；

sessionStorage 的数据在浏览器关闭后自动删除

表单控件，calendar、date、time、email、url、search

新的技术webworker, websocket, Geolocation

移除的元素

纯表现的元素：basefont，big，center，font, s，strike，tt，u；

对可用性产生负面影响的元素：frame，frameset，noframes；

支持HTML5新标签：

IE8/IE7/IE6支持通过document.createElement方法产生的标签，

可以利用这一特性让这些浏览器支持HTML5新标签，

当然最好的方式是直接使用成熟的框架、使用最多的是html5shim框架

<!--[if lt IE 9]>

<script> src="http://html5shim.googlecode.com/svn/trunk/html5.js"</script>

<![endif]-->

如何区分： DOCTYPE声明\新增的结构元素\功能元素

#### **列举IE 与其他浏览器不一样的特性？**

IE支持currentStyle，FIrefox使用getComputStyle

IE 使用innerText，Firefox使用textContent

滤镜方面：IE:filter:alpha(opacity= num)；Firefox：-moz-opacity:num

事件方面：IE：attachEvent：火狐是addEventListener

鼠标位置：IE是event.clientX；火狐是event.pageX

IE使用event.srcElement；Firefox使用event.target

IE中消除list的原点仅需margin:0即可达到最终效果；FIrefox需要设置margin:0;padding:0以及list-style:none

CSS圆角：ie7以下不支持圆角

#### **一个页面从输入 URL 到页面加载显示完成，这个过程中都发生了什么？**

分为4个步骤：

（1），当发送一个URL请求时，不管这个URL是Web页面的URL还是Web页面上每个资源的URL，浏览器都会开启一个线程来处理这个请求，同时在远程DNS服务器上启动一个DNS查询。这能使浏览器获得请求对应的IP地址。

（2）， 浏览器与远程`Web`服务器通过`TCP`三次握手协商来建立一个`TCP/IP`连接。该握手包括一个同步报文，一个同步-应答报文和一个应答报文，这三个报文在 浏览器和服务器之间传递。该握手首先由客户端尝试建立起通信，而后服务器应答并接受客户端的请求，最后由客户端发出该请求已经被接受的报文。

（3），一旦`TCP/IP`连接建立，浏览器会通过该连接向远程服务器发送`HTTP`的`GET`请求。远程服务器找到资源并使用HTTP响应返回该资源，值为200的HTTP响应状态表示一个正确的响应。

（4），此时，`Web`服务器提供资源服务，客户端开始下载资源。

请求返回后，便进入了我们关注的前端模块

简单来说，浏览器会解析`HTML`生成`DOM Tree`，其次会根据CSS生成CSS Rule Tree，而`javascript`又可以根据`DOM API`操作`DOM`

#### **GET和POST的区别，何时使用POST？**

GET：一般用于信息获取，使用URL传递参数，对所发送信息的数量也有限制，一般在2000个字符

POST：一般用于修改服务器上的资源，对所发送的信息没有限制。

GET方式需要使用Request.QueryString来取得变量的值，而POST方式通过Request.Form来获取变量的值，

也就是说Get是通过地址栏来传值，而Post是通过提交表单来传值。

然而，在以下情况中，请使用 POST 请求：

无法使用缓存文件（更新服务器上的文件或数据库）

向服务器发送大量数据（POST 没有数据量限制）

发送包含未知字符的用户输入时，POST 比 GET 更稳定也更可靠

#### **ajax的缺点和在IE下的问题？**

详情请见：[JavaScript学习总结（七）Ajax和Http状态字][14]

ajax的缺点

1、ajax不支持浏览器back按钮。

2、安全问题 AJAX暴露了与服务器交互的细节。

3、对搜索引擎的支持比较弱。

4、破坏了程序的异常机制。

5、不容易调试。

#### **谈谈你对重构的理解**

网站重构：在不改变外部行为的前提下，简化结构、添加可读性，而在网站前端保持一致的行为。也就是说是在不改变UI的情况下，对网站进行优化，   
在扩展的同时保持一致的UI。

对于传统的网站来说重构通常是：

表格(table)布局改为DIV+CSS

使网站前端兼容于现代浏览器(针对于不合规范的CSS、如对IE6有效的)

对于移动平台的优化

针对于SEO进行优化

深层次的网站重构应该考虑的方面

减少代码间的耦合

让代码保持弹性

严格按规范编写代码

设计可扩展的API

代替旧有的框架、语言(如VB)

增强用户体验

通常来说对于速度的优化也包含在重构中

压缩JS、CSS、image等前端资源(通常是由服务器来解决)

程序的性能优化(如数据读写)

采用CDN来加速资源加载

对于JS DOM的优化

HTTP服务器的文件缓存

#### **如何删除一个cookie**

1.将时间设为当前时间往前一点。

var date = new Date();

date.setDate(date.getDate() - 1);//真正的删除

setDate()方法用于设置一个月的某一天。

2.expires的设置

document.cookie = 'user='+ encodeURIComponent('name') + ';expires = ' + new Date(0)

#### **编写一个方法 求一个字符串的字节长度**

假设：一个英文字符占用一个字节，一个中文字符占用两个字节

function GetBytes(str){

var len = str.length;

var bytes = len;

for(var i=0; i<len; i++){

if (str.charCodeAt(i) > 255) bytes++;

}

return bytes;

}

alert(GetBytes("你好,as"));

#### **什么样的前端代码是好的**

高复用低耦合，这样文件小，好维护，而且好扩展。