面试经验

目录

[一、计算机网络/操作系统 2](#_Toc492388139)

[1. socket通信 2](#_Toc492388140)

[2.OSI七层模型和TCP/IP五层模型 2](#_Toc492388141)

[3.TCP/IP协议 3](#_Toc492388142)

[4.浏览器工作原理 3](#_Toc492388143)

[5.Http状态码 5](#_Toc492388144)

[6.TCP和UDP区别 6](#_Toc492388145)

[7.进程和线程区别 6](#_Toc492388146)

[8.Http头信息 6](#_Toc492388147)

[二、前端 7](#_Toc492388148)

[1.mvvm数据双向绑定原理 7](#_Toc492388149)

[2.前端优化十四规则 7](#_Toc492388150)

[3.浏览器如何完成渲染 10](#_Toc492388151)

[4. cookie/localStorage/sessionStorage/userData/session 10](#_Toc492388152)

[三、JS 11](#_Toc492388153)

[1.JS返回数组类型的方法 11](#_Toc492388154)

[3.JSONP原理 11](#_Toc492388155)

[4.this指向 12](#_Toc492388156)

[5.bind实现原理 12](#_Toc492388157)

[6.new一个实例，中间经过哪几个过程 12](#_Toc492388158)

[7.深/浅拷贝 12](#_Toc492388159)

[12.Ajax原理 13](#_Toc492388160)

[15.setTimeout和setInterval原理 14](#_Toc492388161)

[16.js改变css样式 15](#_Toc492388162)

[四、Html/CSS 17](#_Toc492388163)

[2.盒子模型 18](#_Toc492388164)

[五、PHP 19](#_Toc492388165)

[1.单例模式 19](#_Toc492388166)

一、计算机网络/操作系统

1. socket通信

本地通过进程PID唯一标识进程

网络中的进程通过“ip+协议+端口”唯一标识进程

网络进程之间通信使用：UNIX BSD套接字（socket）

socket是在应用层和传输层之间的一个抽象层，它把TCP/IP层复杂的操作抽象为几个简单的接口供应用层调用已实现进程在网络中通信

socket(地址类型，socket类型，协议)、bind(服务器地址，端口号)、listen、connect、accpet、read、write、close

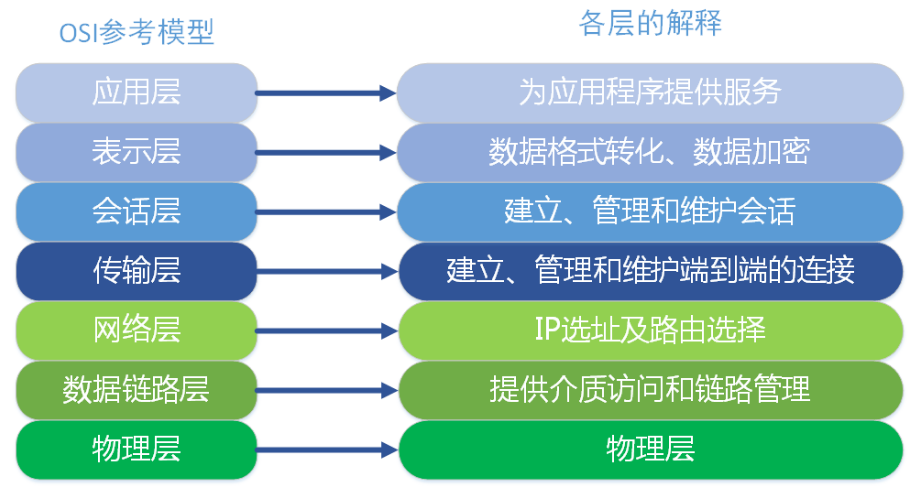
socket类型：

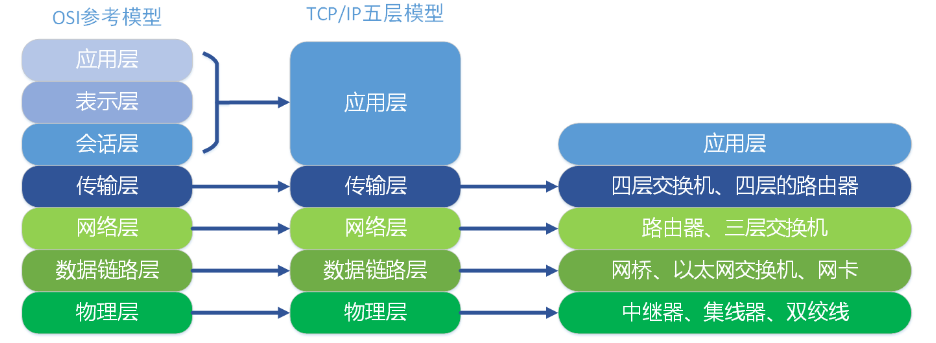
  (1)基于TCP的Socket:提供给应用层可靠的流式数据服务，使用TCP的Socket应用程序协议：BGP，HTTP，FTP，TELNET等。优点：基于数据传输的可靠性。  
   (2)基于UDP的Socket：适用于数据传输可靠性要求不高的场合。基于UDP的Socket应用程序或协议有：RIP，SNMP，L2TP等。  
  (3)基于RawIp的Socket:非连接，不可靠的数据传输。特点：能使应用程序直接访问网络层。基于RawIp的Socket有ping ,tracert,ospf等。  
  (4)基于链路层的Socket。为IS-IS协议提供的Socket接口。使IS-IS协议可通过Socket直接访问链路层。非连接，不可靠通信服务。

<http://www.cnblogs.com/dolphinX/p/3460545.html>

<http://blog.csdn.net/jiajia4336/article/details/8798421>

2.OSI七层模型和TCP/IP五层模型





3.TCP/IP协议

应用层：

tcp/SSH,FTP，HTTP,SMTP,POP3,Telnet

udp/TFTP，SNMP

tcp&udp/DNS

icmp/ping

传输层：TCP，UDP

网络层：IP，ICMP，OSPF，EIGRP，IGMP

数据链路层：SLIP，CSLIP，PPP，MTU

4.浏览器工作原理

**1.域名解析**

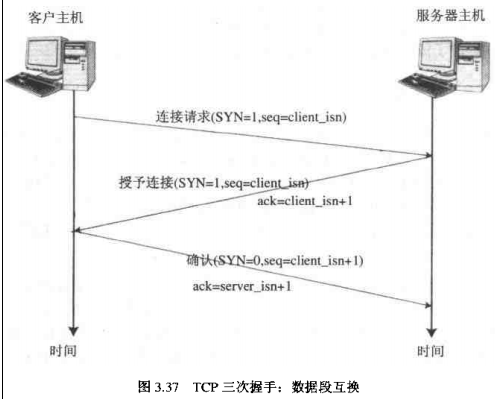
查询浏览器自身DNS缓存

查询操作系统自身DNS缓存

尝试读取hosts文件

向本地配置首选DNS服务器发送请求

**2.与服务器建立通信（TCP三次握手）**



**3.发送http请求**

建立TCP连接后浏览器向服务器发送http请求

浏览器发送其请求命令之后，还要以头信息的形式向Web服务器发送一些别的信息

之后浏览器发送了一空白行来通知服务器，它已经结束了该头信息的发送

**4.服务器处理请求**

服务器软件收到http请求，确定执行什么(ASP.net PHP RUBY JAVA等)来处理它，读取参数并进行逻辑操作后，生成指定的数据

**5.服务器做出应答**

客户机向服务器发出请求后，服务器会客户机回送应答，HTTP/1.1 200 OK ，应答的第一部分是协议的版本号和应答状态码

**6.服务器发送应答头信息**

正如客户端会随同请求发送关于自身的信息一样，服务器也会随同应答向用户发送关于它自己的数据及被请求的文档

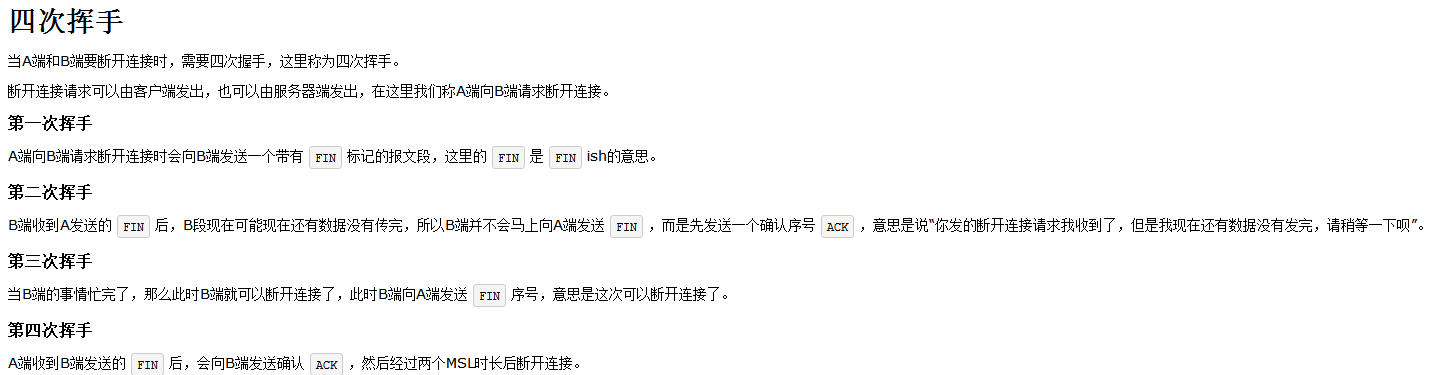
**7.服务器发送数据**

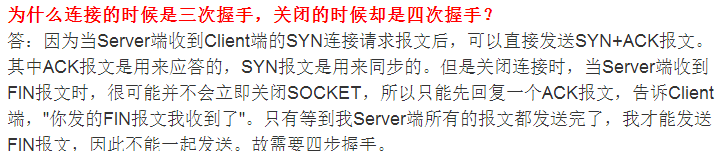
Web服务器向浏览器发送头信息后，它会发送一个空白行来表示头信息的发送到此为结束，接着，它就以Content-Type应答头信息所描述的格式发送用户所请求的实际数据

**8.tcp连接关闭**

一般情况下，一旦Web服务器向浏览器发送了请求数据，它就要关闭TCP连接（四次挥手）

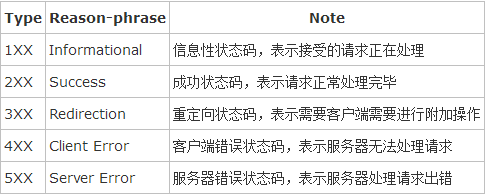
然后如果浏览器或者服务器在其头信息加入了这行代码：Connection:keep-alive   
TCP连接在发送后将仍然保持打开状态，于是，浏览器可以继续通过相同的连接发送请求。保持连接节省了为每个请求建立新连接所需的时间，还节约了网络带宽





▲当tcp第三次握手失败，服务器不会重传报文，而是发送RTS报文段，进入closed状态，防止syn洪泛攻击

5.Http状态码



常用状态码：

**200 OK**  
  一切正常，对GET和POST请求的应答文档跟在后面

**301 Moved Permanently**  
  客户请求的文档在其他地方，新的URL在Location头中给出，浏览器应该自动地访问新的URL。

**302 Found**  
  类似于301，但新的URL应该被视为临时性的替代，而不是永久性的

**304 Not Modified**  
  客户端有缓冲的文档并发出了一个条件性的请求（一般是提供If-Modified-Since头表示客户只想比指定日期更新的文档）。服务器告诉客户，原来缓冲的文档还可以继续使用。

**307 Temporary Redirect**  
  和302（Found）相同。许多浏览器会错误地响应302应答进行重定向，即使原来的请求是 POST，即使它实际上只能在POST请求的应答是303时才能重定向。由于这个原因，HTTP 1.1新增了307，以便更加清楚的区分几个状态代码： 当出现303应答时，浏览器可以跟随重定向的GET和POST请求；如果是307应答，则浏览器只能跟随对GET请求的重定向

**400 Bad Request**  
  请求出现语法错误。

**401 Unauthorized**  
  客户试图未经授权访问受密码保护的页面。应答中会包含一个WWW-Authenticate头，浏览器据此显示用户名字/密码对话框，然后在填写合适的Authorization头后再次发出请求

**403 Forbidden**  
  资源不可用

**404 Not Found**  
  无法找到指定位置的资源

**410 Gone**  
  所请求的文档已经不再可用，而且服务器不知道应该重定向到哪一个地址。它和404的不同在于，返回407表示文档永久地离开了指定的位置，而404表示由于未知的原因文档不可用

**500 Internal Server Error**  
  服务器遇到了意料不到的情况，不能完成客户的请求

**501 Not Implemented**  
  服务器不支持实现请求所需要的功能。例如，客户发出了一个服务器不支持的PUT请求

6.TCP和UDP区别

TCP：

面向连接，三次握手

20字节报头，结构复杂，流模式

数据正确，保证顺序

UDP：

面向报文，传送速度取决于应用程序产生数据速度、计算机能力、传输带宽，接收方将数据放在队列中，每次读取一个消息段

8字节报头，结构简单，数据报模式

不保证数据正确，不保证顺序正确

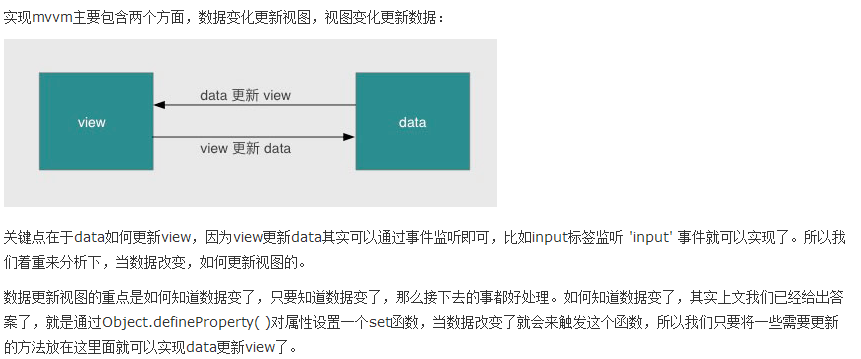
7.进程和线程区别

<http://blog.csdn.net/yaosiming2011/article/details/44280797>

8.Http头信息

二、前端

1.mvvm数据双向绑定原理



2.前端优化十四规则

HTTP是一种客户端/服务器协议，由请求和响应构成。浏览器向一个特定的URL发送HTTP请求，URL对应的宿主服务器发回HTTP响应。该协议使用简单的纯文本格式。请求的类型有GET、POST、HEAD、PUT、DELETE、OPTIONS和TRACE。

**前端性能优化十四个规则：**

**1.      减少HTTP请求。**

a.      至少80%的最终用户响应时间花在了页面中的组件（图片、脚本、样式表、Flash等）上。

b.      改善响应时间的最简单途径就是减少组件的数量，并由此减少HTTP请求的数量。这些技术包括图片地图、CSS Sprites、内联图片和脚本、样式表的合并。运用这些技术在示例页面上估计响应时间减少到50%左右。

c.      图片地图：

优点：允许在一个图片上关联多个URL，目标URL的选择取决于用户点击了图片上的那个位置。这样既能减少HTTP请求，又无需改变页面外观。

缺点：在定义图片地图上的区域坐标时，如果采取手工的方式则很难完成且容易出错，而且除了矩形之外几乎无法定义其他形状。通过DHTML穿件的图片地图则在IE中无法工作。

类型：图片地图有两种类型。服务器端图片地图将所有点击提交到同一个目标URL，向其传递用户单击的x、y坐标。Web应用程序将该x、y坐标映射为适当的操作。客户端图片地图将用户的点击映射到一个操作，而无需向后端应用程序发送请求。映射通过HTML的MAP标签实现。

d.      CSS Sprites

e.      内连图片

f.       合并脚本和样式表

**2.      使用内容发布网络（CDN）**

a.      内容发布网络（CDN）是一组分布在多个不同地理位置的Web服务器，用于更加有效地向用户发布内容。

优点：缩短响应时间、备份、扩展存储能力和进行缓存、缓和Web流量峰值压力。CDN用于发布静态内容，如图片、脚本、样式表和Flash，提供动态HTML页面会引入特殊的存储需求——[**数据库**](http://lib.csdn.net/base/mysql)连接、状态管理、验证、硬件和OS优化等，这些操作比较复杂。另一方面，静态文件更容易存储并具有较少的依赖性。

缺点：响应时间可能会受到其他网站的影响，以及无法直接控制组件服务器所带来的特殊麻烦，且严重依赖CDN的服务性能。

无论如何也不要使用HTTP重定向来将用户指向到本地服务器，这会使Web页面反应速度变慢。

**3.      添加Expires头**

a.      Expires头

浏览器（和代理）使用缓存来减少HTTP请求的数量，并减少HTTP响应的大小，使Web页面加载得更快。Web服务器使用Expires头来告诉Web客户端它可以使用一个组件的当前副本，直到指定的时间为止。它在HTTP响应中发送。

缺点：因为Expires头使用一个特定的时间，它要求服务器和客户端的时钟严格同步。另外，过期日期需要经常检查，并且一旦这一天到来，还需要在服务器配置中提供一个新的日期。

图片可缓存30天以上。

样式表可缓存30天以上。

脚本可缓存30天以上。

b.      Max-Age和mod\_expires

HTTP1.1引入了Cache-Control头来克服Expires头的限制。Cache-Control使用max-age指令指定组件被缓存多久。它以秒为单位定义了一个更新窗。如果从组件被请求开始过去的秒数少于max-age，浏览器就使用缓存的版本，这就避免了额外的HTTP请求。如果同时出现Expires头和Cache-Control max-age，max-age指令将重写Exipres头。

mod\_expires Apache模块，能够让Expires头像max-age那样以相对的方式设置日期。这通过ExpiresDefault指令来完成。

c.      空缓存VS完整缓存

指的是与页面相关的浏览器缓存的状态。

d.      修改文件名

为了确保用户能够获取组件的最新版本，需要在所有HTML页面中修改组件的文件名。嵌入版本号即可。

**4.      压缩组件**

a.      通过减小HTTP响应的大小来减少响应时间。

b.      从HTTP1.1开始，Web客户端可以通过HTTP请求中的Accept-Encoding头来标识对压缩的支持。 Accept-Encoding:gzip,deflate。 如果Web服务器看到请求中有这个头，就会使用客户端列出来的方法中的一种来压缩响应。Web服务器通过响应中的Content-Encoding头来通知Web客户端，如：Content-Encoding: gzip。

c.      网站会压缩其HTML文档、脚本和样式表等。图片和PDF不应该压缩，因为他们本来就已经被压缩了。试图对它们进行压缩只会浪费CPU资源，还有可能会增加文件大小。

d.      压缩的成本有，服务器端会花费额外的CPU周期来完成压缩，客户端要对压缩文件进行解压缩。要检测收益是否大于开销，需要考虑响应的大小、连接的带宽和客户端与服务器之间的Internet距离。

e.      代理缓存。在Web服务器的相应中添加Vary头。Web服务器可以告诉代理根据一个或多个请求头来改变缓存的响应。由于压缩的决定是基于Accept-Encoding请求头的，因此需要在服务器的Vary响应头中包含Accept-Encoding。Vary：Accept-Encoding。

f.       如果拥有大量的、多变的用户群，能够应付较高的带宽开销，并且享有高质量的名声，请压缩内容并使用Cache-Control:Private。这禁用了代理，但避免了边缘情形缺陷。

**5.      将样式表放在顶部**

a.      尽量多使用link标签，少使用@import标签。因为@import标签规则必须放在其他规则之前，也有可能会导致白屏现象，即使把@import规则放在文档的HEAD标签中也是如此。

**6.      将脚本放在底部**

a.      在下载脚本时并行下载实际上是被禁用的——即使使用了不同的主机名，浏览器也不会启动其他的下载。其中一个原因是，脚本可能使用document.write来修改页面内容，因此浏览器会等待，以确保页面能够恰当地布局。另外一个原因是，为了保证脚本能够按照正确的顺序执行。

b.      脚本对Web页面的影响：脚本会阻塞对其后面内容的呈现，以及脚本会阻塞对其后面组件的下载。

//以下规则用于处理页面加载之后的性能问题。

**7.      避免CSS表达式**

//min-width的兼容性

width:expression(document.body.clientWidth<600?”600px”:”auto”);  //IE

min-width:600px;  //大部分浏览器

存在问题：更新表达式

a.      一次性表达式，在这一次执行中重写它自身。

b.      事件处理器。

**8.      使用外部**[**JavaScript**](http://lib.csdn.net/base/javascript)**和CSS**

纯粹而言，内联快一些。但现实中还是使用外部文件会产生较快的页面。因为[**javascript**](http://lib.csdn.net/base/javascript)和CSS文件有机会被浏览器缓存起来。HTML文档通常不会被配置为可以缓存。另外，由于外联，HTML文档减小，HTTP请求的数量也不会增加。

**9.      减少DNS查找**

a.      DNS缓存和TTL(Time-to-live)

DNS查找可以被缓存起来提高性能。服务器可以表明记录可以被缓存多久。查找返回的DNS记录包含了一个存活时间TTL值。该值告诉客户端可以对该记录缓存多久。尽管[**操作系统**](http://lib.csdn.net/base/operatingsystem)缓存会考虑TTL值，但浏览器通常忽略该值，并设置它自己的时间限制。

**10.  精简JavaScript**

a.      精简是从代码中移除不必要的字符以减小其大小，进而改善加载时间的实践。

b.      压缩，并建议使用gzip来完成压缩。

c.      精简CSS。通常CSS中的注释和空白比JavaScript少。优化CSS——合并相同的类、移除不使用的类等。

**11.  避免重定向**

a.      重定向用与将用户从一个URL重新路由到另一个URL。

b.      304并不是真的重定向——它用来响应条件GET请求，避免下载已经存在于浏览器缓存中的数据。

c.      状态码301和302是使用得最多的。状态码303和307是在HTTP1.1规范中添加的。但是几乎没有人用303和307，绝大多数网站仍然在沿用302.301响应的示例如下：

HTTP 1.1 301 Moved Permanently

Location: <http://stevesouders.com/newuri>

Content-Type: text/html

浏览器会自动将用户带到Location字段所给出的URL，响应体通常是空的。不管叫什么名字，301和302响应在实际中都不会被缓存，除非有附加的头——如Expires或Cache-Control等要求它这么做。

d.      重定向延迟了整个HTML文档的传输，在HTML文档到达之前，页面中不会呈现出任何东西，也没人有任何组件会被下载。在用户和HTML文档之间插入重定向延迟了页面中的所有东西。

e.      当缺少结尾的斜线时发送重定向——它允许自动索引，自动转到默认的index.html上，并且能够获得与当前目录相关的URL。

**12.  移除重复脚本**

确保脚本只被包含一次。

**13.  配置ETag**

a.      ETag(Entity Tag)实体标签是Web服务器和浏览器用于确认缓存组件的有效性的一种机制。

b.      ETag是HTTP1.1中引入的。ETag是唯一标识了一个组件的一个特定版本的字符串。唯一的格式约束是该字符串必须用引号引起来。

c.      ETag的加入为验证实体提供了比最新修改日期更为灵活的机制。

d.      例如，如果实体依据User-Agent或Accpt-Language头而改变，实体的状态可以反映到ETag中。此后，如果浏览器必须验证一个组件，它会使用If-None-Match头将ETag传回原始服务器。如果ETag是匹配的，就会返回304状态码，使响应减小了1195字节。

**14.  使Ajax可缓存**

确保Ajax请求遵守性能知道，尤其应具有长久的Expires头。

3.浏览器如何完成渲染

1、IE下载的顺序是从上到下，渲染的顺序也是从上到下，下载和渲染是同时进行的。  
2、在渲染到页面的某一部分时，其上面的所有部分都已经下载完成（并不是说所有相关联的元素都已经下载完）。  
3、如果遇到语义解释性的标签嵌入文件（JS脚本，CSS 剑 敲创耸盜E的下载过程会启用单独连接进行下载。  
4、并且在下载后进行解析，解析过程中，停止页面所有往下元素的下载。  
5、样式表在下载完成后，将和以前下载的所有样式表一起进行解析，解析完成后，将对此前所有元素（含以前已经渲染的）重新进行渲染。  
6、JS、CSS中如有重定义，后定义函数将覆盖前定义函数

4. cookie/localStorage/sessionStorage/userData/session

▲session由服务器存储数据，当客户端浏览器关闭，session释放

▲cookie由浏览器本地存储小于4k（总量）的数据，但是需要传入服务器处理再传回浏览器

▲localStorage html5新技术，由浏览器本地直接存储小于5M（总量）的数据，不需要经过服务器，存储格式为string，浏览器隐私模式下不可读取，localStorage本质上是对字符串的读取，如果存储内容多的话会消耗内存空间，会导致页面变卡，localStorage不能被爬虫抓取到

▲**localStorage的写入**有三种方法

var storage= localStorage;

//写入a字段

storage["a"]=1;

//写入b字段

storage.a=1;

//写入c字段

storage.setItem("c",3);

读取：官方推荐的是getItem\setItem这两种方法对其进行存取

修改：storage.a=4;

清空：storage.clear();

删除：storage.removeItem("a");

对象的localStorage存储需要先通过JSON转化为字符串，JSON.stringify，提取则用JSON.parse

▲localStorage与sessionStorage的唯一一点区别就是localStorage属于永久性存储，而sessionStorage属于当会话结束的时候，sessionStorage中的键值对会被清空

sessionStorage.lastname="Smith";

5.三列布局，左右固定宽度，中间自适应

（1）先左右后中间

思路一：

左float：left，右float：right

中直接放入，width：100%，margin

思路二：

左position：absolute，left：0

右position：absolute，right：0

中直接放入，width：100%，margin

（2）两翼法

思路一：

中float：left，width：100%

左float：left，margin：-100%

右float：left，margin：-宽度

中box-sizing：border-box，padding

思路二：

中float：left，width：100%

左float：left，margin：-100%

右float：left，margin：-宽度

中再嵌套一个div，设置该div的margin

（3）flex

建一个父容器

#par{

display: flex;

flex-direction:row;

}

按顺序放置三个容器

#middle{

width:100%;

height:100px;

background:yellow;

}

#left{

width:200px;

height:100px;

background: red;

}

#right{

width:400px;

height:100px;

background: green;

}

6.观察者模式

7.实现函数，每调用一次状态改变

8.class继承

9.css实现图片轮播淡入淡出

10.事件代理

三、JS

1.JS返回数组类型的方法

typeof：识别基础类型，funtion可以输出”function”，null会输出”object”

Object.prototype.toString.apply()：在ie6/7/8会把null和undefined识别为”[object Object]”

instanceof：在不同iframe和window之间失效，返回true/false

isArray：Array对象的方法

constructor：返回对象的构造函数

2.跨域的方式有几种

（1）代理

因为同源策略是针对客户端的，在服务器端没有什么同源策略，是可以随便访问的，所以我们可以通过下面的方法绕过客户端的同源策略的限制：

客户端先访问同源的服务端代码，该同源的服务端代码

使用httpclient等方法，再去访问不同源的服务端代码，然后将结果返回给客户端，这样就间接实现了跨域。

相关例子，参见博文：http://www.cnblogs.com/digdeep/p/4198643.html

（2）cors

在服务端开启cors也可以支持浏览器的跨域访问

cors即：Cross-Origin Resource Sharing 跨域资源共享

jsonp和cors的区别是jsonp几乎所有浏览器都支持，但是只能是get

而cors有些老浏览器不支持，但是get/post都支持

**（3）PHP端修改header（XHR2方式）**

在php接口脚本中加入以下两句即可：

header('Access-Control-Allow-Origin:\*');//允许所有来源访问

header('Access-Control-Allow-Method:POST,GET');//允许访问的方式

（4）web Workers

3.JSONP原理

ajax请求受浏览器同源策略影响，不允许进行跨域请求，而script标签src属性中的链接却可以访问跨域的js脚本，利用这个特性，服务端不再返回JSON格式的数据，而是返回一段调用某个函数的js代码，在src中进行了调用，这样实现了跨域

4.this指向

（1）全局this和普通函数中的this表示window

（2）对象方法中的this表示调用该方法的对象

（3）对象方法中独立函数（如setTimeout中的匿名函数）的this表示window

（4）构造器中的this表示新创建的实例对象

（5）DOM事件绑定函数中的this表示该DOM元素

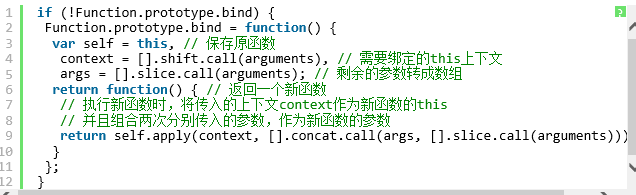
（6）箭头函数中的this在定义时就已经绑定，与上下文中this指向相同

5.bind实现原理

call，apply临时绑定this，直接调用函数

bind生成一个与原函数一样的绑定this和实参函数，返回这个新函数

当bind()所返回的函数用作构造函数的时候，传入bind()的this将被忽略，但实参会全部传入原函数



6.new一个实例，中间经过哪几个过程

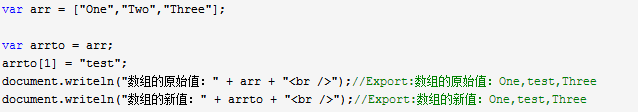
（1）建立一个新对象

（2）将新对象的\_\_proto\_\_指向构造函数的prototype

（3）绑定构造函数的this为新对象，执行构造函数，返回新对象

7.深/浅拷贝

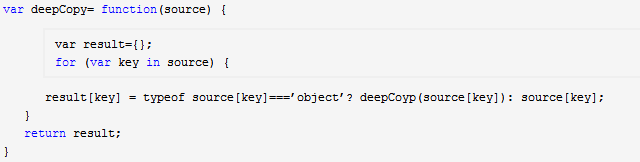
**数组浅拷贝**：直接将数组名赋值给其他变量时，赋给该变量的只是一个对象的引用，当改变数组时，两个变量也将改变



数组深拷贝：通过方法对数组操作，生成新的数组对象再赋值给一个变量

如：arr.slice(0); arr.concat();

对象深拷贝：建立一个新的对象，将原对象属性都赋给新对象



8.webpack脚手架

9.原型链

10.闭包

11.作用域链

12.Ajax原理

判断是否支持XMLHttpRequest，有则创建一个新的XMLHttpRequest对象，ie5/6不支持则创建一个新的ActiveXObject对象

if ( window.XMLHttpRequest ){

xmlhttp = new XMLHttpRequest();

}else{

xmlhttp = new ActiveXObject(“Microsoft.XMLHTTP”);

}

发送Ajax请求

xmlhttp.open(“GFT/POST”,url,true/false);

xmlhttp.send();

Ajax响应

xmlhttp.onreadystatechange = function(){

if (xmlhttp.readystate==4 && xmlhttp.status==200){

xmlhttp.responseText/xmlhttp.responseXML  
}

}

13.DOM结点增删查改

元素节点的 nodeName 是标签名称

属性节点的 nodeName 是属性名称

文本节点的 nodeName 永远是 #text

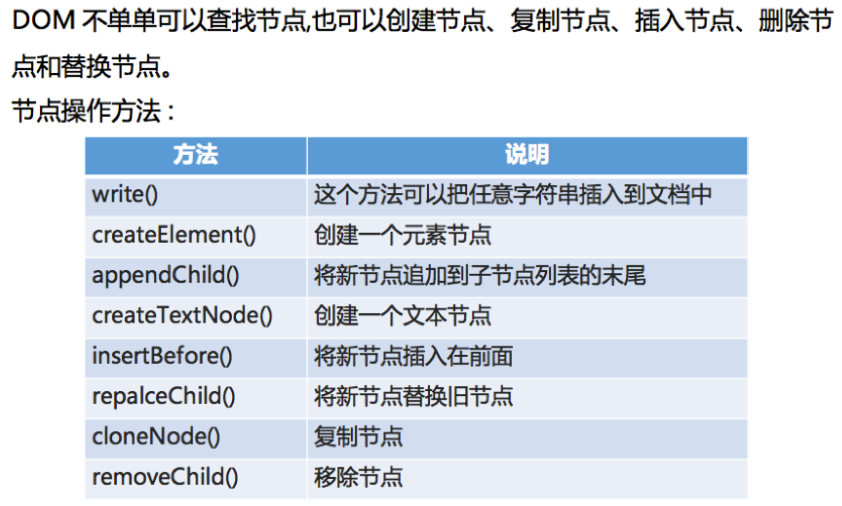
文档节点的 nodeName 永远是 #document

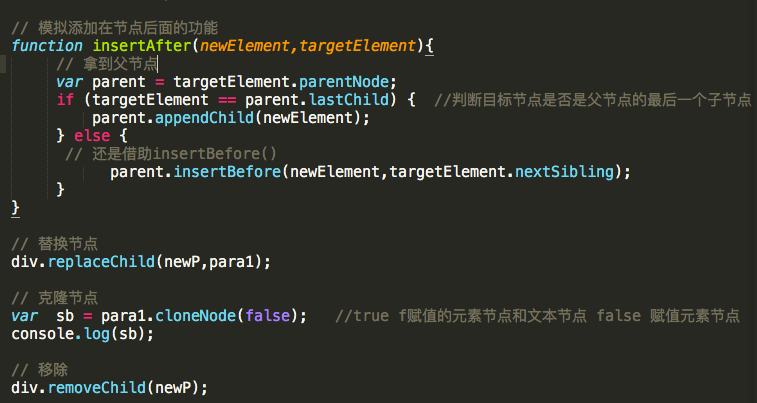
对于文本节点，nodeValue 属性包含文本。

对于属性节点，nodeValue 属性包含属性值。

nodeValue 属性对于文档节点和元素节点是不可用的。







14.js内存泄漏的操作

内存泄露是指一块被分配的内存既不能使用，又不能回收，直到浏览器进程结束

（1）当页面中元素被移除或替换时，若元素绑定的事件仍没被移除，在IE中不会作出恰当处理，此时要先手工移除事件，不然会存在内存泄露

（2）在 Internet Explorer 中，如果循环引用中的任何对象是 DOM 节点或者 ActiveX 对象，垃圾收集系统则不会发现它们之间的循环关系与系统中的其他对象是隔离的并释放它们。最终它们将被保留在内存中，直到浏览器关闭

（3）闭包可以维持函数内局部变量，使其得不到释放。

解决之道：

减少使用闭包

或者将闭包作用域中没用的属性删除，以减少内存消耗

（4）删除对象，如果对象的属性被引用

在JavaScript的某些实现中，在销毁对象的时候，要遍历属性中属性，依次删除

15.setTimeout和setInterval原理

setTimeout(func,time)：

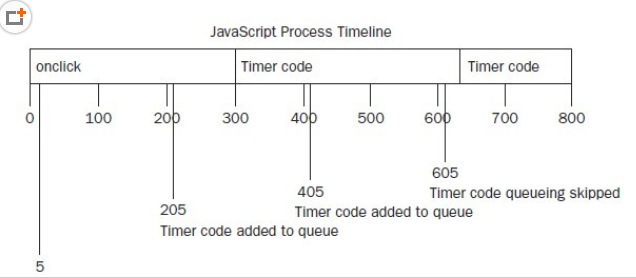
在时间time后将func代码段插入到js代码队列中，如果time时间在执行另一个代码段，则需要等这个代码段执行完毕再执行func代码段

setInterval(func,time)：

（1）有可能跳过时间间隔

（2）时间间隔可能小于time

同样在time时间后，js代码队列插入func代码段，等正在的程序执行完之后，执行func代码段，time是从开始计时的，如果func执行时间过长，2\*time时间后又一个func插入队列，如果3\*time时间后第2个func还未执行，那么第3个func会被跳过，因为js引擎只允许一个未执行的func



一种解决方案：

setTimeout(function(){   
//processing   
setTimeout(arguments.callee, interval);   
}, interval);

当func程序执行完后再setTimeout

16.js改变css样式

obj = document.getElementById(“btn”)

（1）修改单个嵌入式样式

▲obj.style.textAlign = “center”;

如果属性有'-'号，就写成驼峰的形式（如textAlign）  如果想保留 - 号，就中括号的形式  element.style['text-align'] = 'center';

▲obj.setAttribute('height', '100px');

（2）修改整体嵌入式样式

▲obj.style.cssText = “background-color:black; display:block; color:White;”

▲obj.setAttribute('style', 'height: 100px !important');

（3）修改class名

▲obj.className = “btnA”;

▲obj.setAttribute(“class”,”btnA”);

（4）更改使用的外联css文件

▲<link href="css1.css" rel="stylesheet" type="text/css" id="css"/>

function changeStyle4() {

   var obj = document.getElementById("css");

   obj.setAttribute("href","css2.css");

 }

（5）使用setProperty

如果要设置!important，推荐用这种方法设置第三个参数

element.style.setProperty('height', '300px', 'important');







四、Html/CSS

1.垂直居中方法

（1）单行

{

height:100px;

line-height:100px;

over-flower:hidden;

}

（2）不定高

{

padding-top:10px;

padding-bottom:10px;

}

（3）表格单元

{

display:table;  
}

{

display:table-cell;

height:50px;

text-align:center;

}

img{

vertical-align:center;  
}

（4）绝对定位

{

position:relative;

}

{

position:absolute;

top:50%;

margin-top:-25px;

}

或者

{

position:absolute;

top/bottom/left/right:0;

}

（5）flex

{

display:flex;

align-items;center;

justify-content:center;

}

（6）transform

{

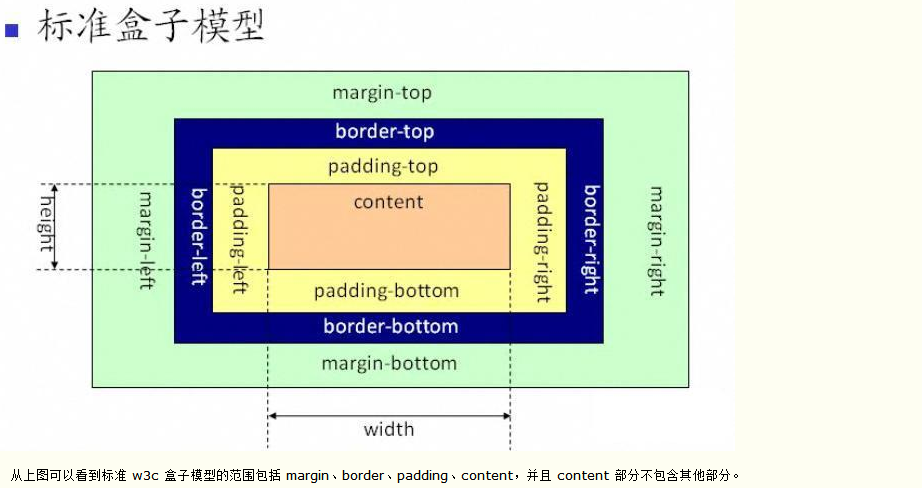
position:absolute;

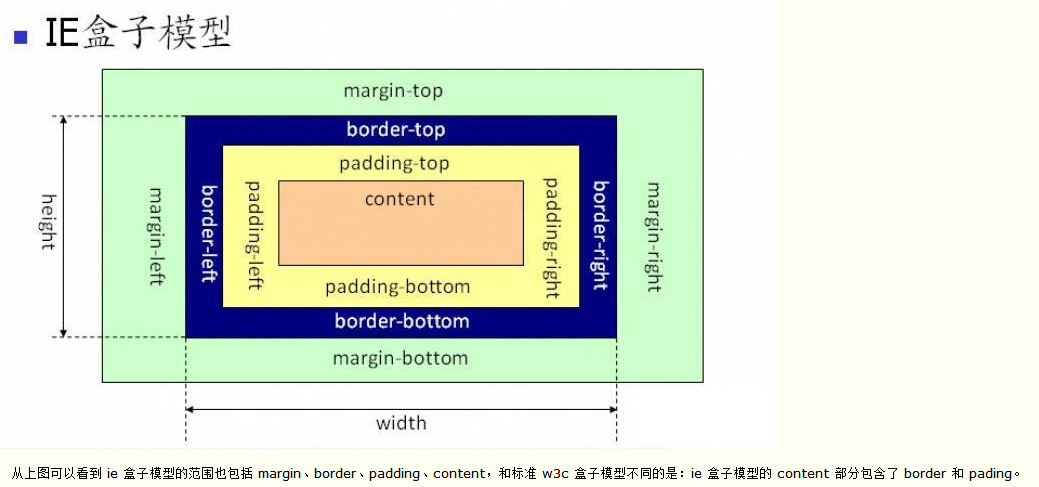
top:50%;

transform:translate(left,-50%);

}

2.盒子模型





3.flex弹性布局

4.BootStracp栅格布局



col-xs- col-sm- col-md- col-lg-

五、PHP

1.单例模式