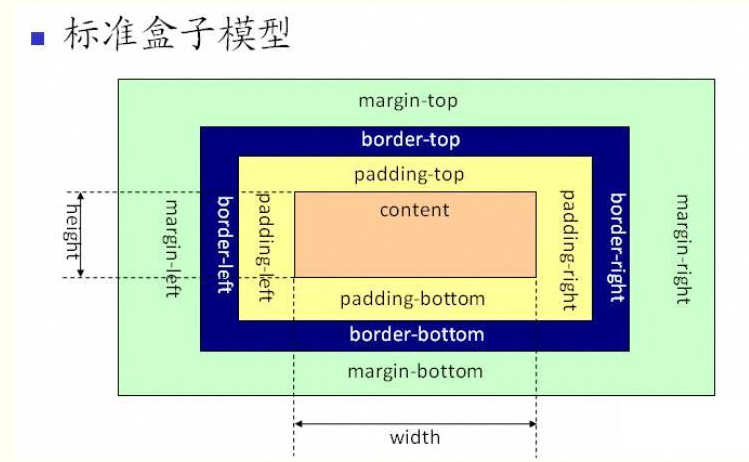
## 1盒模型

标准盒模型 ：content不包含padding，margin。

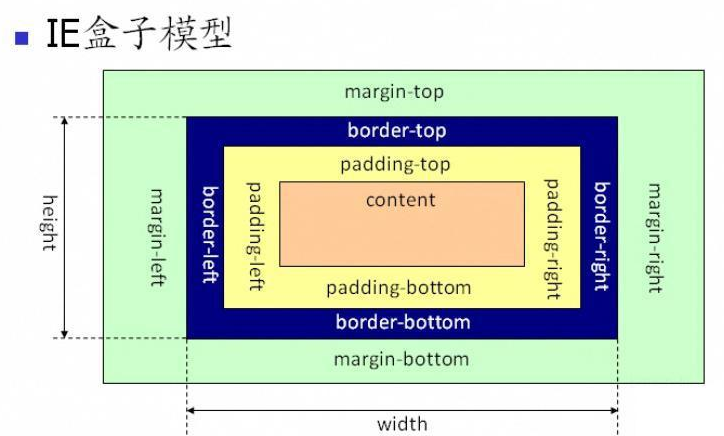
ie盒子模型：content 包含padding和border。

**因此计算盒子宽度**

标准 =（padding+margin）\*2+border+width



IE = margin\*2+width



## 2关于websocket

websocket一种基于tcp，全双工，一次握手，持久连接的协议

老的替代物有

Ajax long-polling技术（ajax长轮询）

flash技术。

在支持WebSocket的浏览器中，在创建socket之后。可以通过onopen，onmessage，onclose即onerror四个事件实现对socket进行响应

一个简单示例

var ws = new WebSocket(“ws://localhost:8080”);

ws.onopen = function()

{

  console.log(“open”);

  ws.send(“hello”);

};

## 3关于webworker

实例化 Worker 对象并传入要执行的 Javascript 文件名就可以创建一个新的 Web Worker：

1. var worker = new Worker('worker.js');

这段代码会导致浏览器下载 worker.js，但只有 Worker 接收到消息才会实际执行文件中的代码。要给 Worker 传递消息，可以使用 postMessage() 方法

主程序js和webworkerjs通信都是用postmessage（）;

* Webworker是单独作用域的，无法访问到window 对象
* document 对象
* parent 对象

## 4 JS基本数据类型和引用数据类型

基本数据类型：Number，String，Boolean，Null，Undefined

引用数据类型: array，function，object

## 5 html5语义化理解

方便人和虫的阅读

## 6 缓存

## Manifest 文件

manifest 文件是简单的文本文件，它告知浏览器被缓存的内容（以及不缓存的内容）。

manifest 文件可分为三个部分：

·        ***CACHE MANIFEST*** - 在此标题下列出的文件将在首次下载后进行缓存

·        ***NETWORK*** - 在此标题下列出的文件需要与服务器的连接，且不会被缓存

·        ***FALLBACK*** - 在此标题下列出的文件规定当页面无法访问时的回退页面（比如 404 页面）

Cache manifest：这下面部分会缓存

network：不会缓存

fallback：无法访问时回退页面

## 7cookie与webstorage

cookie缺陷：

1. 数据大小只有4kb左右
2. 安全性问题，在http请求中是明文传递的
3. 每次http请求都会带上
4. 不同浏览器还有数量限制

Webstorage

不是替代cookie，cookie是为了客户端与服务端通信，

有时候会被用来客户端本地存储，。webstorage为了解决这个问题

webstorage分，session storage和localstorage

一个是永久储存，sessionstorage是会话期间有效。

    sessionStorage：不在不同的浏览器窗口中共享，即使是同一个页面；

     localStorage：在所有同源窗口都是共享的；

     cookie：也是在所有同源窗口中共享的

拓展：webStorage只能操作**字符串对象**

## 8多页面通信

cookie，web worker，webstorage和windows.name等

## 9 键入url到页面渲染的过程。

1输入地址

2浏览器查找域名ip地址，（浏览器缓存，系统缓存，路由器缓存）

3浏览器向web服务器发送一个http请求。

4服务器重定向

5浏览器跟踪重定向

6服务器处理请求

7服务器返回http请求

8解析html文档，生成树

9渲染css，请求嵌入在html的资源

10浏览器发送异步请求

## 10重构、回流

浏览器的重构指的是改变每个元素外观时所触发的浏览器行为，比如颜色，背景等样式发生了改变而进行的重新构造新外观的过程。重构不会引发页面的重新布局，不一定伴随着回流，

回流指的是浏览器为了重新渲染页面的需要而进行的重新计算元素的几何大小和位置的，他的开销是非常大的，回流可以理解为渲染树需要重新进行计算，一般最好触发元素的重构，避免元素的回流；比如通过通过添加类来添加css样式，而不是直接在DOM上设置，当需要操作某一块元素时候，最好使其脱离文档流，这样就不会引起回流了，比如设置position：absolute或者fixed，或者display：none，等操作结束后在显示

## 11 JavaScript的闭包

闭包简单的说就是一个函数能访问外部函数的变量

## 12new操作符干了什么？

新建一个空对象，把this指向该对象

空对象的原型指向函数原型，

最后改变构造函数内部this指向

## 13call和apply和bind

改变函数内部this指向可以fn.call(obj);或者fn.apply(obj)

,然后立即执行，第一个参数都是对象，第二个apply为数组，

call为arg1，arg2，arg3这种。

另外bind只有一个参数，就是对象。

返回一个函数，

exp： var bar=fn.bind（obj）;

## 14js的作用域和作用域链

js原来无块级作用域只有函数作用域和全局作用域，es6有了let。

内部作用域和外部的作用域一层层的链接起来形成作用域链，当在在函数内部要访问一个变量的时候，首先查找自己的内部作用域有没有这个变量，如果没有就到这个对象的原型对象中去查找，还是没有的话，就到该作用域所在的作用域中找，直到到window所在的作用域

## 15回去啃完js高程那部分之后再来做补充理解

## 16变量提升

var bar=1;

function test(){

console.log(bar); //undeifned

var bar=2;

console.log(bar); //2

}

test();

为什么在test函数中会出现上述结果呢，这就是JavaScript的变量提升了，虽然变量bar的定义在后面，不过浏览器在解析的时候，会把变量的定义放到最前面，上面的test函数相当于

function test(){

var bar;

console.log(bar); //undefined

**bar=2;**

console.log(bar); //2

}

再看

**var foo=function(){ console.log(1); }**

**function foo(){ console.log(2); } //函数定义**

**foo(); //结果为1**

同样的，函数的定义也会到提升到最前面，上面的代码相当于

**function foo(){ console.log(2); }**

**var foo;**

**foo=funciton(){ console.log(1); }**

**foo(); //1**

## 17、、

**浏览器端：**

cookie

WebStorage(localStorage、sessionStorage)

userData

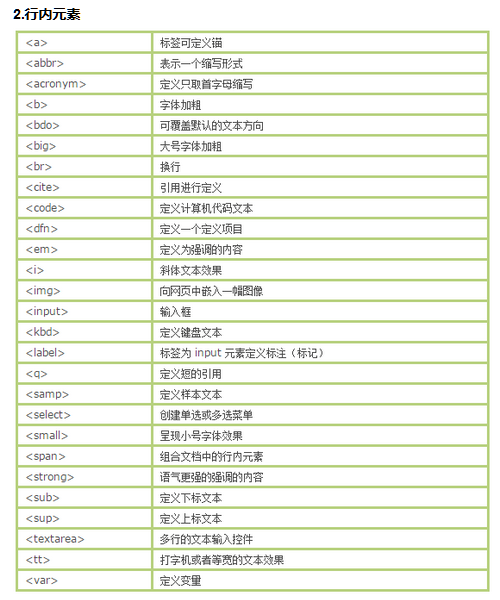
indexedDB

**服务器端：**

session

## 18块级行内元素





## 19内存泄漏

内存泄漏指的是浏览器不能正常的回收内存的现象

浏览器的垃圾回收机制

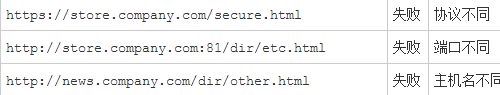
垃圾收集器必须跟踪哪个变量有用哪个变量没用，对于不再有用的变量打上标记，以备将来收回其占用的内存，内存泄露和浏览器实现的垃圾回收机制息息相关， 而浏览器实现标识无用变量的策略主要有下两个方法：

1. **引用计数法 第二，标记清除法**

## 20 同源策略 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。

## 21跨域方案分析

## http://www.cnblogs.com/2050/p/3191744.html 非同源失败案例

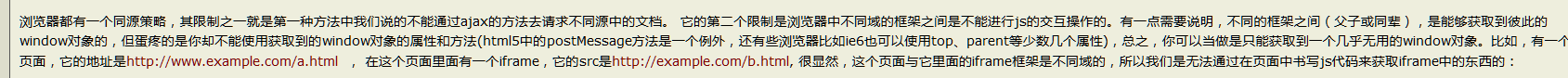


#### 方案1 jsonp

xhr无法取得不同域上数据是因为同源限制，但是页面上引入js脚本是可以的



#### 方案2修改document.domain跨子域





修改子域只能一层，修改了也不能ajax

**3、使用window.name来进行跨域**

很方便。直接改写名字，但是实现比较丑陋。

**4、使用HTML5中新引进的window.postMessage方法来跨域传送数据**

window.postMessage(message,targetOrigin)  方法是html5新引进的特性，可以使用它来向其它的window对象发送消息，无论这个window对象是属于同源或不同源，目前IE8+、FireFox、Chrome、Opera等浏览器都已经支持window.postMessage方法。

调用postMessage方法的window对象是指要接收消息的那一个window对象，该方法的第一个参数message为要发送的消息，类型只能为字符串；第二个参数targetOrigin用来限定接收消息的那个window对象所在的域，如果不想限定域，可以使用通配符 \*  。

需要接收消息的window对象，可是通过监听自身的message事件来获取传过来的消息，消息内容储存在该事件对象的data属性中。

上面所说的向其他window对象发送消息，其实就是指一个页面有几个框架的那种情况，因为每一个框架都有一个window对象。在讨论第二种方法的时候，我们说过，不同域的框架间是可以获取到对方的window对象的，而且也可以使用window.postMessage这个方法。下面看一个简单的示例，有两个页面



## 22异步和同步

同步指下一个程序的执行需要等到上一个程序执行完毕，也就是得出结果后下一个才能执行，

异步指的是上一个程序指向后，下一个程序不用等到上一个程序出结果就能执行，等上一个出结果了调用回调函数处理结果就好。

## 23 JavaScript的值类型和引用类型（深拷贝和浅拷贝）

问题对数组对象的复制，因为是引用类型，复制的只是内存指针指向

## 24array方法整理

1.**检测是否为数组**

If（value instanceof Array）;(在假定只有一个全局作用域情况下)

因此在es5中增加方法，Array.isArray();返回boolean

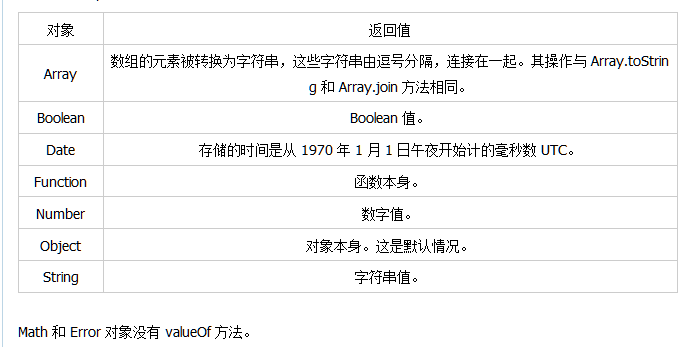
2．**转换方法**

1 toString()，所有对象都拥有的方法，把对象转为字符串，对于数组，可以把各个数组每一项拼接起来成为字符串。

2 tolocalstring()，把tolocalstring，同tostring区别在于对每个数组项调用的是toloaclstring（）方法

toLocalString()方法，把Date对象转为字符串，并根据本地规则规范化。

3 valueof()



### 2栈和队列方法

Pop数组尾移除

Push数组尾添加

Shift数组头移除

Unshift：数组头添加

### 3重排序方法

Reverse(); 颠倒顺序

Sort()默认升序，【sort会调用每个数组项的tostring方法，即使是数字也会转为字符串，因此不合适】

**比如 "123" 和 "3" 比较 其实比较的是 "1" 和 "3"的 ascii码**

要自己写比较函数，比较函数接受两个参数

（如果第一个参数应该位于第二个后面返回一个负数，相等返回0.）

Exp： function compare(value1,value2){

If(value1>value2)

Return 1;

Else if(value1<value2)

Return -1;

Else return 0;}

从大到小排列的比较函数

}

### 4操作方法

1 **concat** 拼接，参数为数组或者非数组，非数组会直接添加到数组尾 一般用’+’，不添加参数时直接复制当前数组返回新数组

也是这个方法返回的为新创建数组，不会改变原数组，因此可以用他来完成数组深拷贝。

2 **slice** 剪切，返回当前数组的一个或者多个项，默认返回从第一项到末尾项，也可以两个参数，起始位置和结束位置，若结束位置>起始位置，则返回空数组。若参数为负，则加上长度。

另外这个方法返回的为新创建数组，不会改变原数组，因此可以用他来完成数组深拷贝。

2.1 深拷贝与浅拷贝

**在使用JavaScript对数组进行操作的时候，我们经常需要将数组进行备份，事实证明如果只是简单的将它赋予其他变量，那么我们只要更改其中的任何一个，然后其他的也会跟着改变，这就导致了问题的发生。**

**1 数组深复制{concat深复制&slice深复制}**

**Var arr1=new Array()；**

**arr2=arr1.concate=()；**

**arr3=arr1.slice();**

**2 对象的深复制**

**就是把所有属性都复制一遍**

**,如果内部还是obj则递归调用**

**Function deepcopy（obj）{**

**Var resultobj={};**

**for（var key in obj）**

**{ resultobj[key]=typeof obj[key]==’object’?deepsource(obj[key]): obj[key];**

**}**

**Return resultobj;**

**}**

**==&&===**

**两个对象，数组比较比较的是内存地址，因此即使属性相同也返回false**

**3最强大的数组方法，splice**

splice方法第一个参数，要删除数的位置，第二个参数为要删除多少项，后面的参数为删除完后添加进去的项

因此 第二个参数若为0，则在第一个参数位置插入后面的项

第二个参数若为不为0，而后面没有参数了，则为删除操作

替换则为第二参数为n，后面跟着n个项

简而言之：可以向指定位置**删除任意数量**的项，在**添加任意数量的项，**

**同样返回新数组**

## 5位置方法

5.1 indexof从前开始找，

lastindexof从后开始找，

返回查找项在数组中的位置。

2参数要**查找的项，起始位置索引（可选）**

## 6迭代方法

每个方法都接受两个参数，要在每一项运行的函数和（可选）函数的作用域对象。【改变this的指向】

分别是**every(),filter(),forEach(),map(),some()**

其中**every**和**some**都是用做查询数组中的项**是否满足条件**，some只要有一项为true才返回true，every要全部为true才返回true。

第一个参数函数都是fuction(item,index,array){

}**接受的参数为，数组项的值，数组项的索引，数组本身**

**forEach() 对数组每一项运行某函数，没有返回值。**

**filter和map都是返回一个新数组；**

**filter把符合函数条件的项组成新数组返回。**

**map，类似foreach，对每一项运行某函数，返回处理后项组成的新数组**

## 7归并方法

reduce，reduceRight从前开始归并，和从后开始归并

第一个参数为函数，接受四个参数（前一个值，当前值，项的索引，数组对象），这个函数返回任何值会作为第一个参数传给下一项，因此第一次迭代发生在数组第二项。

# 25 [js中typeof与instanceof用法小记](http://www.cnblogs.com/cheng_jiai/archive/2010/10/07/1845168.html)

**typeof用以获取一个变量或者表达式的类型，typeof一般只能返回如下几个结果：**

**number,boolean,string,function（函数）,object（NULL,数组，对象）,undefined。**

**instanceof用于判断一个变量是否某个对象的实例，如var a=new Array();alert(a instanceof Array);会返回true，同时alert(a instanceof Object)也会返回true;这是因为Array是object的子类。再如：function test(){};var a=new test();alert(a instanceof test)会返回true。**

# 26 ==和===

==值相同，===值和类型全相同。

## 27 判断js对象是否含有某属性

1 in运算符

返回boolean，可以获得原型链上的属性

exp：

var obj={name:’wjm’};

console.log(‘name’ in obj)//’true’

console.log(‘toString’ in obj)//’true’

2hasOwnPropertry

Var obj={name:’wjm’};

obj.hasOwnPropetry(‘name’);//ture

obj.hadOwnPropetry(‘toString’)//false

## 28元素脱离文档流导致的高度塌陷问题

**1.问题描述：**  
在文档流中，父元素的高度默认是被子元素撑开的，也就是子元素多高，父元素就多高。但是当子元素设置浮动之后，子元素会完全脱离文档流，此时将会导致子元素无法撑起父元素的高度，导致父元素的高度塌陷。

## 29事件委托与事件代理

<http://www.cnblogs.com/liugang-vip/p/5616484.html>

写的很棒

## 30http请求状态码

http详细解释http://www.cnblogs.com/ranyonsue/p/5984001.html

**http状态码**

**http状态码是表示服务器对请求的响应状态，主要分为以下几个部分**

**1\*\*：这类响应是临时响应，只包含状态行和某些可选的响应头信息，并以空行结束**

**2\*\*：表示请求成功，**

**3\*\*：表示重定向**

**4\*\*：表示客户端错误**

**5\*\*：表示服务器端错误**

**100（continue），客户端应当继续发送请求。这个临时响应是用来通知客户端它的部分请求已经被服务器接收**

**200（OK），表示请求成功，请求所希望的响应头或数据体将随此响应返回。**

**202（Accepted），服务器已接受请求，但尚未处理。**

**204（No-Content），服务器成功处理了请求，但不需要返回任何实体内容**

**205（Reset-Content），服务器成功处理了请求，且没有返回任何内容。但是与204响应不同，返回此状态码的响应要求请求者重置文档视图。该响应主要是被用于接受用户输入后，立即重置表单，以便用户能够轻松地开始另一次输入。**

**206（Partial-Content），服务器已经成功处理了部分 GET 请求。**

**301（Moved-Permanently），永久性重定向**

**302（Moved-Temporarily），暂时性重定向**

**304（Not-Modified），浏览器端缓存的资源依然有效**

**400（Bad-Reques），请求有误，当前请求无法被服务器理解。**

**401（Unauthorized），当前请求需要用户验证。**

**403（Forbidden），服务器已经理解请求，但是拒绝执行它。**

**404（Not-Found），请求的资源没有被找到**

**500（Interval Server Error），服务器内部错误**

**502（Bad GateWay），网关出错**

**503（Service Unavailable），由于临时的服务器维护或者过载，服务器当前无法处理请求。**

**504（Gateway Timeout），作为网关或者代理工作的服务器尝试执行请求时，未能及时从上游服务器（URI标识出的服务器，例如HTTP、FTP、LDAP）或者辅助服务器（例如DNS）收到响应。**

## 31 xss，csrf的概念以及防范方法

<http://blog.csdn.net/ghsau/article/details/17027893>

<http://www.cnblogs.com/hyddd/archive/2009/04/09/1432744.html>

## 32 前端模块化

https://zhuanlan.zhihu.com/p/22954387

## 33五层协议

## 34再次cookie。Loaclstorage，sessionstorage

## 

## 35 <http://www.tuicool.com/articles/UjMNFfB>

## 36懒加载原理

http://www.cnblogs.com/flyromance/p/5042187.html

**如1）屏幕可视窗口大小：对应于图中1、2位置处**

**原生方法：window.innerHeight 标准浏览器及IE9+ || document.documentElement.clientHeight 标准浏览器及低版本IE标准模式 ||**

**document.body.clientHeight 低版本混杂模式**

**jQuery方法： $(window).height()**

**2）浏览器窗口顶部与文档顶部之间的距离，也就是滚动条滚动的距离：也就是图中3、4处对应的位置；**

**原生方法：window.pagYoffset——IE9+及标准浏览器 || document.documentElement.scrollTop 兼容ie低版本的标准模式 ||**

**document.body.scrollTop 兼容混杂模式；**

**jQuery方法：$(document).scrollTop();**

**3）获取元素的尺寸：对应于图中5、6位置处；左边jquery方法，右边原生方法**

**$(o).width() = o.style.width;**

**$(o).innerWidth() = o.style.width+o.style.padding;**

**$(o).outerWidth() = o.offsetWidth = o.style.width+o.style.padding+o.style.border;**

**$(o).outerWidth(true) = o.style.width+o.style.padding+o.style.border+o.style.margin;**

**注意：要使用原生的style.xxx方法获取属性，这个元素必须已经有内嵌的样式，如<div style="...."></div>；**

**如果原先是通过外部或内部样式表定义css样式，必须使用o.currentStyle[xxx] || document.defaultView.getComputedStyle(0)[xxx]来获取样式值**

**4）获取元素的位置信息：对应与图中7、8位置处**

**1）返回元素相对于文档document顶部、左边的距离；**

**jQuery：$(o).offset().top元素距离文档顶的距离，$(o).offset().left元素距离文档左边缘的距离**

**原生：getoffsetTop()，高程上有具体说明，这边就忽略了；**

　 　　  顺便提一下返回元素相对于第一个以定位的父元素的偏移距离，注意与上面偏移距的区别；

    　　  jQuery：position()返回一个对象，$(o).position().left = style.left，$(o).position().top = style.top；

## 果是图片，动态设置，src

# 37@media 查询

## 

## 38全屏滚动原理

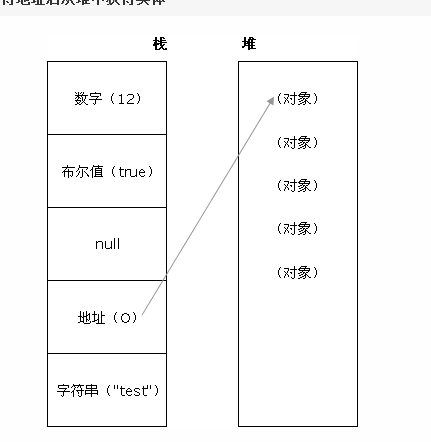
获取窗口高度，window.height

距离最上面高度 var scrollTop = document.body.scrollTop || document.documentElement.scrollTop || window.pageYOffset;;

## 39原型，原型链



## 40 基本数据类型，和引用数据类型内存图



## 41Javascript如何实现继承？

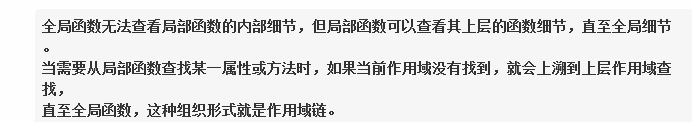


## 42js创建对象方式

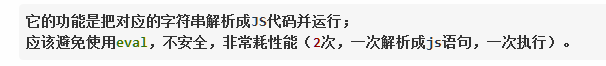




## 43js作用域链



## 44eval是做什么的？

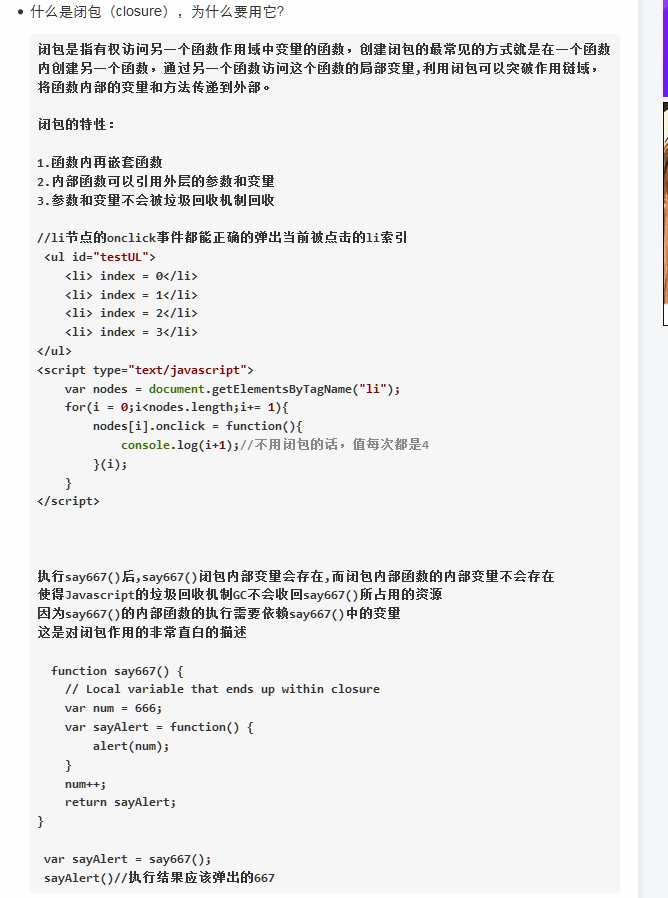


## 45null，undefined 的区别？

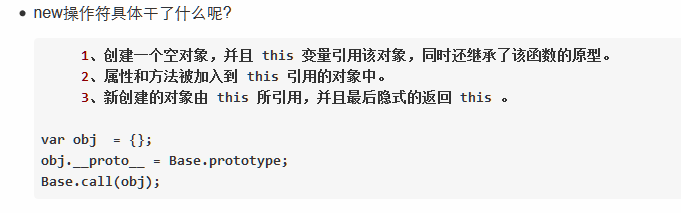


46通用的事件监听函数

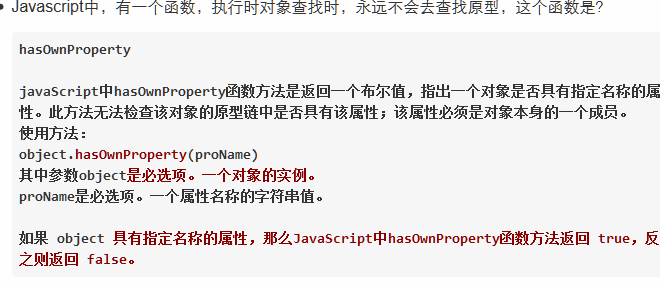
## 47再次闭包



## 47new操作符再临



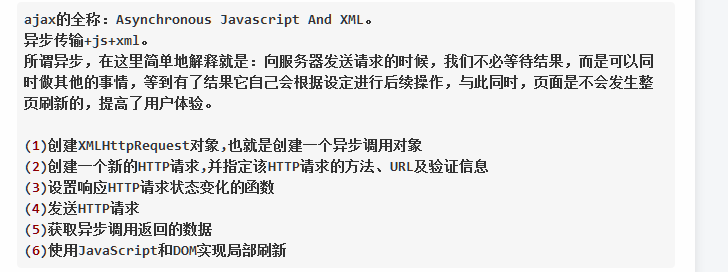
## 48hasOwnprotetry再临



## 49js延迟加载的方式有哪些？

defer和async、动态创建DOM方式（用得最多）、按需异步载入js

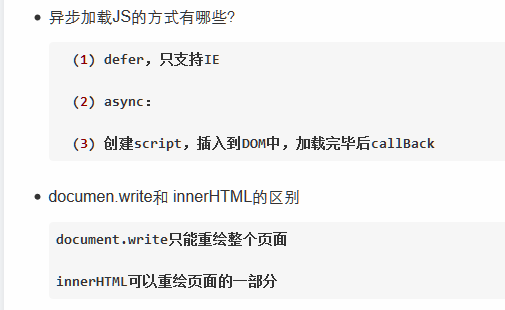
## 50AJAX再临



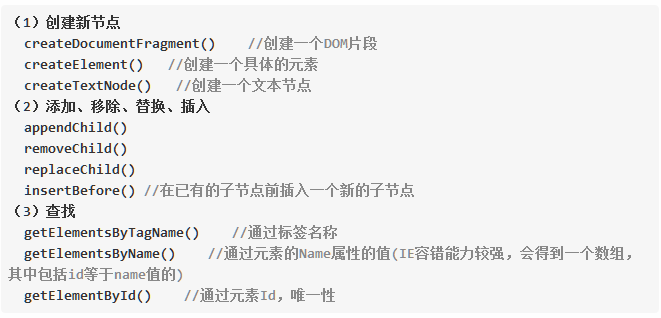
## 51跨域再临



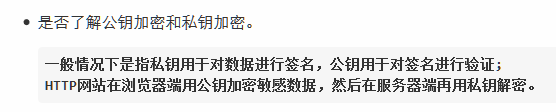
## 52细碎1



## 53DOM操作——怎样添加、移除、移动、复制、创建和查找节点?



## 54公钥私钥

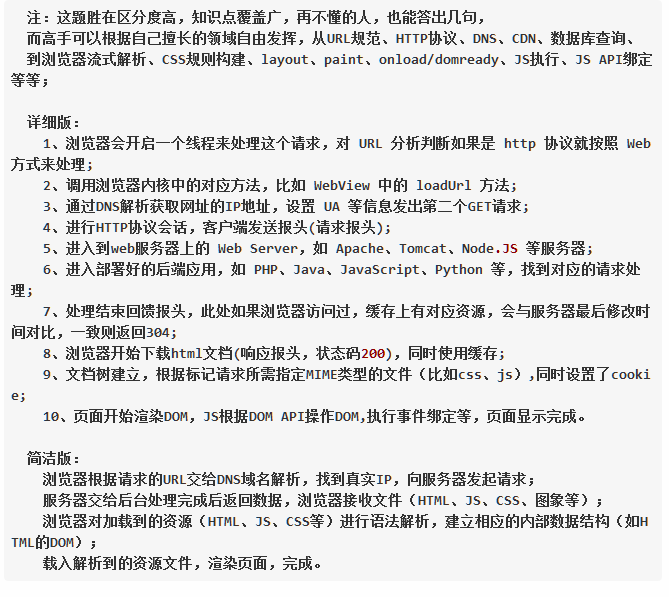


## 55对Node的优点和缺点提出了自己的看法？

## 56前端性能优化



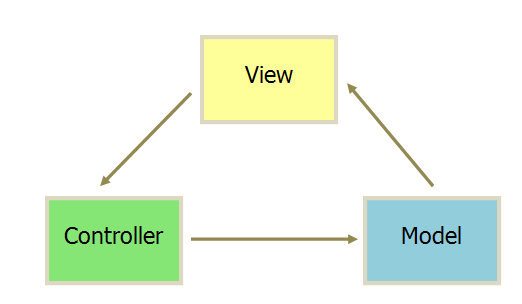
## 57 一个页面从输入 URL 到页面加载显示完成，这个过程中都发生了什么？（流程说的越详细越好）



## 58

## mvc架构

Model,View,Controller



### Mvvm

### Mvp

## 59前后端分离以及ajax对seo的影响

一种解决方案

http://www.ruanyifeng.com/blog/2013/07/how\_to\_make\_search\_engines\_find\_ajax\_content.html

## 60nodejs 作为中间件渲染html作用

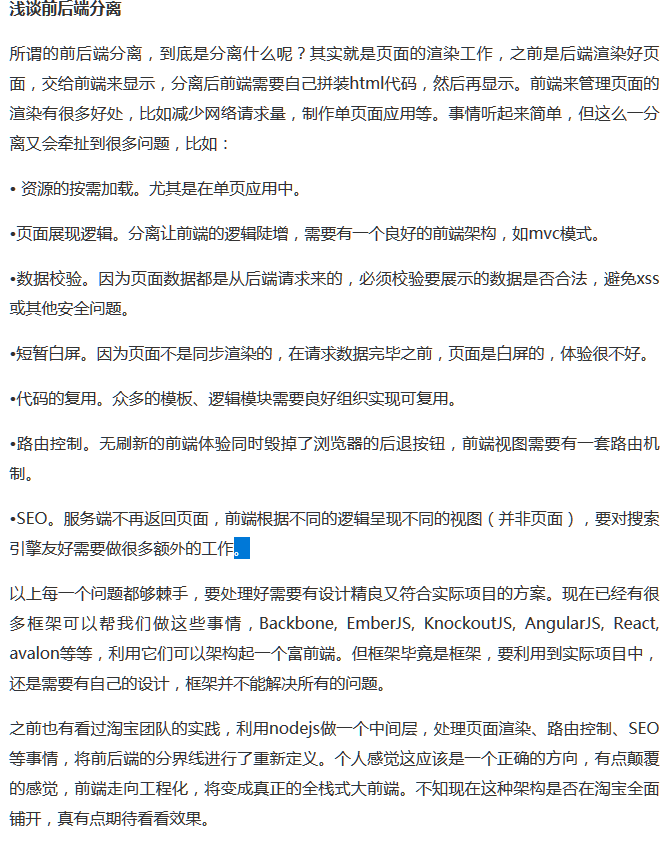
Seo优化方案2：Ajax会影响seo，而用node.js作为中间件渲染出前端html利于seo

从根本解决seo问题

Seo优化方案3 老的那种静态写法。

## 61再谈前后端分离

## 参考淘宝团队实践：一个简单粗暴的前后端分离方案<http://developer.51cto.com/art/201501/463338.htm>



**前端渲染nodejs的中间件作用实例**

**nodejs做一个中间层，处理页面渲染、路由控制、SEO等事情，将前后端的分界线进行了重新定义**

## 62 spa中前进后退按钮的实现

原链接<http://www.tuicool.com/articles/BZvqQn>

产生原因，ajax不会刷新页面，因此url会保持不变，而为了安全考虑history对象不能增删改，只能调用其原生方法history.go(),history.back(),history.forward();

因此要通过改变url的变化来进行一对一的映射。

而url每次变化加载会产生一个历史纪录扔到浏览器的历史堆栈中，通过window.location.hash属性读写。

那怎么生成这些url呢？就在页面中放一个隐藏的iframe，然后通过html5 history **api**进行修改

**html5 history API**





# 63懒加载核心终于找到了，判断页面是否在可视区

**js getBoundingClientRect()方法用法介绍:  
此函数可以获取指定对象的左上右下分别相对于浏览器窗口的位置。  
它可以返回一个对象，此对象具有四个属性left、top、right和bottom分别能够返回各自方位相对于浏览器窗口的尺寸。  
返回值的单位是像素(px)。  
语法结构:**

**[JavaScript] *纯文本查看* *复制代码运行代码***

**oRect = object.getBoundingClientRect()；**

|  |  |
| --- | --- |
|  |  |
| 64window.onload和document.ready Windows.onload是页面加载完，  Document.ready 是jquery封装的，猜测下是dom树加载完成时 65html5 media事件 |  |