HTML+CSS

**1.对WEB标准以及W3C的理解与认识**  
标签闭票合、标签小写、不乱嵌套、提高搜索机器人搜索几率、使用外链css和js脚本、结构行为表现的分离、文件下载与页面速度更快、内容能被更多的用户所访问、内容能被更广泛的设备所访问、更少的代码和组件，容易维 护、改版方便，不需要变动页面内容、提供打印版本而不需要复制内容、提高网站易用性；  
**2.xhtml和html有什么区别**  
HTML是一种基本的WEB网页设计语言，XHTML是一个基于XML的置标语言  
最主要的不同：  
XHTML 元素必须被正确地嵌套。  
XHTML 元素必须被关闭。  
标签名必须用小写字母。  
XHTML 文档必须拥有根元素。  
  
3.Doctype? 严格模式与混杂模式-如何触发这两种模式，区分它们有何意义?   
用于声明文档使用那种规范（html/Xhtml）一般为 严格 过度 基于框架的html文档  
加入XMl声明可触发，解析方式更改为IE5.5 拥有IE5.5的bug  
  
4.行内元素有哪些?块级元素有哪些?CSS的盒模型?

块级元素：div p h1 h2 h3 h4 form ul  
行内元素: a b br i span input select  
Css盒模型:内容，border ,margin，padding  
  
5.CSS引入的方式有哪些? link和@import的区别是?  
内联 内嵌 外链 导入  
区别 ：同时加载  
前者无兼容性，后者CSS2.1以下浏览器不支持  
Link 支持使用javascript改变样式，后者不可

6.CSS选择符有哪些?哪些属性可以继承?优先级算法如何计算?内联和important哪个优先级高?  
标签选择符 类选择符 id选择符  
继承不如指定 Id>class>标签选择  
后者优先级高

7.前端页面有哪三层构成，分别是什么?作用是什么?  
结构层 Html 表示层 CSS 行为层 js

9

8.css的基本语句构成是?  
选择器{属性1:值1;属性2:值2;……}  
  
9.你做的页面在哪些流览器测试过?这些浏览器的内核分别是什么?  
Ie(Ie内核) 火狐（Gecko） 谷歌（webkit） opear(Presto)  
  
10.写出几种IE6 BUG的解决方法  
1.双边距BUG float引起的 使用display  
2.3像素问题 使用float引起的 使用dislpay:inline -3px

**bug描述**：ie下两个相邻的div之间会出现3个像素的bug，这个bug是在当对其中一个div使用了float，而另外一个没有使用时会出现。

**解决方法：**a.对另一个元素同时使用float； b.为**已经浮动的div**添加一条语句：margin-right:-3px; ；

3.超链接hover 点击后失效 使用正确的书写顺序 link visited hover active  
4.Ie z-index问题 给父级添加position:relative  
5.Png 透明 使用js代码 改  
6.Min-height 最小高度 ！Important 解决’  
7.select 在ie6下遮盖 使用iframe嵌套  
8.为什么没有办法定义1px左右的宽度容器（IE6默认的行高造成的，使用over:hidden,zoom:0.08 line-height:1px）  
（1）超链接点击过后hover样式就不出现？

解答：被点击访问过的超链接样式不再具有hover和active样式了,解决方法是改变CSS属性的排列顺序: L-V-H-A

（2）IE6的margin双倍边距bug问题

解答：<style type='text/css'>

body{margin:0px}

div{float:left;margin-left:10px;width:200px;}

</style>

浮动后本来外边距10px,但IE解释为20px,解决办法是加上display:inline

（3）为什么中火狐浏览器下文本无法撑开容器的高度？

解答：标准浏览器中固定高度值的容器是不会象IE6里那样被撑开的,那我又想固定高度，又想能被撑开需要怎样设置呢？办法就是去掉height设置min-height:200px; 这里为了照顾不认识min-height的IE6 可以这样定义：div { height:auto!important; height:200px; min-height:200px; }

（4）为什么web标准中无法设置IE浏览器滚动条颜色了？

解答：原来样式设置：解决办法是将body换成html

<style type='text/css'>

body { scrollbar-face-color:#f6f6f6; scrollbar-highlight-color:#fff; scrollbar-shadow-color:#eeeeee; scrollbar-3dlight-color:#eeeeee; scrollbar-arrow-color:#000; scrollbar-track-color:#fff; scrollbar-darkshadow-color: #fff; }

</style>

（5）如何定义1px左右高度的容器？

解答：IE6下这个问题是因为默认的行高造成的，解决的方法也有很多，例如：overflow:hidden | zoom:0.08 | line-height:1px

（6）怎样使一个div层居中于浏览器中？

style type='text/css'>

<!--

div {

position:absolute;

top:50%;

left:50%;

margin:-100px 0 0 -100px;

width:200px;

height:200px;

border:1px solid red;

}

-->

</style>

这里使用百分比绝对定位，与外补丁负值的方法，负值的大小为其自身宽度高度除以二

Flex版

.parent {

display:flex;

align-items: center;/\*垂直居中\*/

justify-content: center;/\*水平居中\*/

width:100%;

height:100%;

background-color:red;

}

.children {

background-color:blue;

}

HTML：

<div class="parent">

<div class="children">我是通过flex的水平垂直居中噢！</div>

</div>

Css3:

.container {

position: absolute;

top: 50%;

left: 50%;

transform: translate(-50%, -50%); // 自身宽度和高度的一半

border: 1px solid red;

}

（7）firefox浏览器中嵌套div标签的居中问题的解决方法

假定有如下情况：

<div id='a'>

<div id='b'> </div>

</div>

　　如果要实现b在a中居中放置，一般只需用CSS设置a的text-align属性为center。这样的方法在IE里看起来一切正常；但是在Firefox中b却会是居左的。

　解决办法就是设置b的横向margin为auto。例如设置b的CSS样式为：margin: 0 auto;。

11.标签上title与alt属性的区别是什么?  
Alt 当图片不显示是 用文字代表。  
Title 为该属性提供信息  
  
12.描述css reset的作用和用途。  
Reset重置浏览器的css默认属性 浏览器的品种不同，样式不同，然后重置，让他们统一  
  
13.解释css sprites，如何使用。  
Css 精灵 把一堆小的图片整合到一张大的图片上，减轻服务器对图片的请求数量  
  
14.浏览器标准模式和怪异模式之间的区别是什么?  
盒子模型 渲染模式的不同  
使用 window.top.document.compatMode 可显示为什么模式  
  
15.你如何对网站的文件和资源进行优化?期待的解决方案包括：  
文件合并  
文件最小化/文件压缩  
使用CDN托管  
缓存的使用  
  
16.什么是语义化的HTML?  
直观的认识标签 对于搜索引擎的抓取有好处  
  
17.清除浮动的几种方式，各自的优缺点  
1.使用空标签清除浮动 clear:both（理论上能清楚任何标签，，，增加无意义的标签）  
2.使用overflow:auto（空标签元素清除浮动而不得不增加无意代码的弊端,,使用zoom:1用于兼容IE）  
3.是用afert伪元素清除浮动(用于非IE浏览器)  
  
**Javascript**  
1.javascript的typeof返回哪些数据类型  
Object number function boolean underfind  
  
2.例举3种强制类型转换和2种隐式类型转换?  
强制（parseInt,parseFloat,number）  
隐式（== – ===）  
  
3.split() join() 的区别  
前者是切割成数组的形式，后者是将数组转换成字符串  
  
4.数组方法pop() push() unshift() shift()  
Push()尾部添加 pop()尾部删除  
Unshift()头部添加 shift()头部删除  
  
5.事件绑定和普通事件有什么区别  
普通事件中的onclick是DOM0级事件只支持单个事件，会被其他onclick事件覆盖，而事件绑定中的addEventListener是DOM2级事件可以添加多个事件而不用担心被覆盖，孰优孰略，聪明的你一定知道了吧。  
6.IE和DOM事件流的区别  
1）.执行顺序不一样、  
2）.参数不一样  
3）.事件加不加on  
4）.this指向问题

<button id="btn">button</button>

<button id="btn2">button2</button>

<script>

var btn = document.getElementById('btn');

btn.onclick = function() {

alert('普通事件1'); //不执行

}

btn.onclick = function() {

alert('普通事件2'); //弹出

}

//上面用普通方法添加两个事件，下面用事件绑定添加两个事件。

btn.addEventListener('click', function() {

alert('普通事件1');

}, false)

btn.addEventListener('click', function() {

alert('普通事件2');

}, false)

//注意我这里只使用W3C的标准写法添加事件，没有兼容低版本的IE。

//注册事件

function addEventHandler(element, evtName, callback, useCapture) {

if (element.addEventListener) { //DOM标准

element.addEventListener(evtName, callback, useCapture);

} else { //IE方式，忽略useCapture

element.attachEvent('on' + evtName, callback);

}

}

//移除

function addremoveEventHandler(element, evtName, callback, useCapture) {

if (element.removeEventListener) { //DOM标准

element.removeEventListener(evtName, callback, useCapture);

} else { //IE方式，忽略useCapture

element.dettachEvent('on' + evtName, callback);

}

}

</script>  
跨浏览器的停止事件传递的方法是:

1.**function** someHandle(event) {

2.  event = event || window.event;

3.  **if**(event.stopPropagation)

4.     event.stopPropagation();

5.  **else**  event.cancelBubble = **true**;

6.}

//跨浏览器的取消事件传递后的默认处理方法是：

function someHandler(event) {

event = event || window.event

if (event.preventDefault) {

event.preventDefault();

} else {

event.returnValue = false

}

}  
7.IE和标准下有哪些兼容性的写法  
Var ev = ev || window.event  
document.documentElement.clientWidth || document.body.clientWidth  
Var target = ev.srcElement||ev.target  
  
8.ajax请求的时候get 和post方式的区别  
一个在url后面 一个放在虚拟载体里面  
有大小限制, GET方式提交的数据最多只能是1024字节  
安全问题, GET方式请求的数据会被浏览器缓存起来  
应用不同 一个是论坛等只需要请求的，一个是类似修改密码的

若符合下列任一情况，则用POST方法：

\* 请求的结果有持续性的副作用，例如，数据库内添加新的数据行。

\* 若使用GET方法，则表单上收集的数据可能让URL过长。

\* 要传送的数据不是采用7位的ASCII编码。

若符合下列任一情况，则用GET方法：

\* 请求是为了查找资源，HTML表单数据仅用来帮助搜索。

\* 请求结果无持续性的副作用。

\* 收集的数据及HTML表单内的输入字段名称的总长不超过1024个字符。  
9.call和apply的区别  
Object.call(this,obj1,obj2,obj3)  
Object.apply(this,arguments)  
  
10.ajax请求时，如何解释json数据  
使用eval parse 鉴于安全性考虑 使用parse更靠谱  
  
12.写一个获取非行间样式的函数  
//获取行间样式函数

function getStyle(dom, attr) {

if (dom.currentStyle) { //如果符合就tru

return dom.currentStyle[attr];

} else {

return getComputedStyle(dom, false)[attr];

}

}

//测试

window.onload = function() {

var btn = document.getElementById('btn')

console.log(getStyle(btn, 'width'))

}  
13.事件委托是什么  
让利用事件冒泡的原理，让自己的所触发的事件，让他的父元素代替执行！  
http://www.webasily.com/?p=78 例子可见此链接  
//事件委托示例

window.onload = function() {

var oUl = document.getElementById('ul1');

oUl.onclick = function(ev) {

var ev = ev || window.event;

var target = ev.target || ev.srcElement;

if (target.nodeName.toLowerCase() == 'li') {

console.log(target.innerHTML);

//如果点击每一个做不同的操作，那么至少需要4次dom操作，如果用事件委托，能进行优化吗？

switch (target.id) {

case 'add':

console.log('添加')

break;

case 'delete':

console.log('删除')

break;

case 'move':

console.log('移动')

break;

case 'select':

console.log('选择')

break;

}

}

}

}

适合用事件委托的事件：click，mousedown，mouseup，keydown，keyup，keypress。  
**14.闭包是什么，有什么特性，对页面有什么影响**我的理解是，闭包就是能够读取其他函数内部变量的函数。

由于在Javascript语言中，只有函数内部的子函数才能读取局部变量，因此可以把闭包简单理解成“定义在一个函数内部的函数”。

所以，在本质上，闭包就是将函数内部和函数外部连接起来的一座桥梁  
闭包可以用在许多地方。它的最大用处有两个，一个是前面提到的可以读取函数内部的变量，另一个就是让这些变量的值始终保持在内存中。

function f1(){

　　　　var n=999;

　　　　nAdd=function(){n+=1}

　　　　function f2(){

　　　　　　alert(n);

　　　　}

　　　　return f2;

　　}

　　var result=f1();

　　result(); // 999

　　nAdd();

　　result(); // 1000  
15.如何阻止事件冒泡和默认事件  
canceBubble return false  
function stopBubble(e) {

//如果提供了事件对象，则这是一个非IE浏览器

if ( e && e.stopPropagation )

    //因此它支持W3C的stopPropagation()方法

    e.stopPropagation();

else

    //否则，我们需要使用IE的方式来取消事件冒泡

    window.event.cancelBubble = true;

}

//阻止浏览器的默认行为

function stopDefault( e ) {

    //阻止默认浏览器动作(W3C)

    if ( e && e.preventDefault )

        e.preventDefault();

    //IE中阻止函数器默认动作的方式

    else

        window.event.returnValue = false;

    return false;

}

16.添加 删除 替换 插入到某个接点的方法  
obj.appendChidl()  
obj.innersetBefore  
obj.replaceChild  
obj.removeChild

<**ul** id = **"fruit"**>  
 <**li** id = **"apple"**>苹果</**li**>  
 <**li** id = **"orange"**>橘子</**li**>  
 <**li**>西瓜</**li**>  
 <**li**>梨</**li**>  
</**ul**>

**var *d1*** = **document**.getElementById(**"d1"**);  
*//节点的创建***var *frNode*** = **document**.getElementById(**'fruit'**);  
**var *liNode*** = **document**.createElement(**'li'**);  
**var *hmgNode*** = **document**.createTextNode(**'哈密瓜'**);  
*//节点的添加****liNode***.appendChild(***hmgNode***);  
***frNode***.appendChild(***liNode***);  
*//节点的替换***var *appleNode*** = **document**.getElementById(**"apple"**);  
***h*** = **document**.createElement(**"li"**);  
**h**.**innerHTML**=**"<span style='color:green'> 这是一个被h1标记包含的 span 标签</span>"**;  
***frNode***.replaceChild(**h**,***appleNode***);  
*//删除***if**(**document**.getElementById(**'orange'**)){  
 ***frNode***.removeChild(**document**.getElementById(**'orange'**))  
}

17.解释jsonp的原理，以及为什么不是真正的ajax  
ajax请求受同源策略影响，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。

18.javascript的本地对象，内置对象和宿主对象  
本地对象为array obj regexp等可以new实例化  
内置对象为gload Math 等不可以实例化的  
宿主为浏览器自带的document,window 等  
  
19.document load 和document ready的区别  
Document.onload 是在结构和样式加载完才执行js  
Document.ready原生种没有这个方法，jquery中有 $().ready(function)  
  
20.”==”和“===”的不同  
前者会自动转换类型  
后者不会  
  
21.javascript的同源策略  
一段脚本只能读取来自于同一来源的窗口和文档的属性，这里的同一来源指的是主机名、协议和端口号的组合

**Js 延时加载的方法：**

当一个网站有很多js代码要加载，js代码放置的位置在一定程度上将会影像网页的加载速度，为了让我们的网页加载速度更快，总结了一下几个Js 延时加载的方法：

1.使用setTimeout延迟方法的加载时间

2.让js最后加载3.把js外部引入的文件放到页面底部，来让js最后引入，从而加快页面加载速度。  
**Position 定位：**

1，relative（相对定位）的定位原点是以自己本身原来所在位置做为原点的。  
2，absolute（绝对定位）的定位原点是离自己这一级元素最近的一级position设置为absolute或者relative的父元素的左上角为原点的，（当然，如果自己的所有父元素都没有设置position，那么就以body为定位原点）

**浏览器不同页面之间的传值:**1. Get方式，url携带参数

可以通过window.location.search获取url上的参数。如下面的示例。

a.HTML：

<a href="./b.html?a=1&a=2&username=zhangsan&password=123&=">跳转</a>

b.html：

<script type="text/javascript" src="./js/getUrlParam.js"></script>

<script>

　　var a = UrlParam.paramValues("a");

　　console.log(a);

</script>

2、通过cookie，传递

cookie能够存储少量数据到客户端的磁盘中，特定的网页之间是可以共享cookie中的数据。

a.html：

<script type="text/javascript" src="./js/jquery.min.js"></script>

<script type="text/javascript" src="./js/jquery.cookie.js"></script>

<script>

　　$.cookie("a","12");

</script>

b.html：

<script type="text/javascript" src="./js/jquery.min.js"></script>

<script type="text/javascript" src="./js/jquery.cookie.js"></script>

<script>

　　var param = $.cookie("a");

　　console.log(param);

</script>

3、h5技术，window.localStorage存储数据

a.html

<input type="text" name="username" />

<input type="button" name="" value="post" onclick="set()"/>

<script>

function set(){

if(window.localStorage){

//存储变量的值

localStorage.name = document.all.username.value;

location.href = './b.html';

}

}

</script>

b.html

<script>

var value = localStorage["name"];

console.log(value);

</script>

4.sessionStorage存储

**var *userEntry*** = {  
 **name**: **'tom'**,  
 **age**:12  
}  
*//存储值，将对象转换为Json字符串*sessionStorage.setItem(**'user'**, JSON.stringify(***userEntry***));  
*//取值：把获取的Json字符串转换为对象***var *userJsonStr*** = sessionStorage.getItem(**'user'**);  
***userEntry*** = JSON.parse(***userJsonStr***)  
console.log(***userEntry***.**name**); *// => tom*

**什么是Json**  
Json一种轻量级的数据交换格式.  
它是基于JavaScript的一个子集，数据格式简单，易于读写，占用带宽小  
例如:{"age":"12","name":"back"}  
JSON 字符串转换为Json对象  
     var obj = eval('('+str+')');  
     var obj = str.parseJSON();  
     var obj = JSON.parse(str);  
JSON 对象转换为JSON字符串  
     var last = obj.toJSONString();  
     var last = JSON.stringify(obj);  
  
**什么Ajax,如何创建Ajax?**  
     ajax的全称: Asynchronous Javascript And XML  
     异步传输+js+xml  
     所谓异步,在这里简单的解释就是:向服务器发送请求的时候,我们不必等待结果,而是可以同时做其他的事情,等到有了结果它自己会根据设定进行后续操作,与此同时,页面是不会发送整夜刷新的,提高了用户体验.  
     (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  
$.ajax({

url: '/XXXX/XXXXX',

dataType: 'json',

crossDomain: true == !(document.all),

contentType: "application/json",

type: 'get',

success: function (data) {}

}),

**1.双边距BUG float引起的 使用display**2.3像素问题 使用float引起的 使用dislpay:inline -3px  
3.超链接hover 点击后失效 使用正确的书写顺序 link visited hover active  
4.Ie z-index问题 给父级添加position:relative  
5.Png 透明 使用js代码 改  
6.Min-height 最小高度 ！Important 解决’  
7.select 在ie6下遮盖 使用iframe嵌套  
8.为什么没有办法定义1px左右的宽度容器（IE6默认的行高造成的，使用over:hidden,zoom:0.08 line-height:1px）  
**1. CSS3有哪些新特性？**1. CSS3实现圆角（border-radius），阴影（box-shadow），  
2. 对文字加特效（text-shadow、），线性渐变（gradient），旋转（transform）  
3.transform:rotate(9deg) scale(0.85,0.90) translate(0px,-30px) skew(-9deg,0deg);// 旋转,缩放,定位,倾斜  
4. 增加了更多的CSS选择器  多背景 rgba  
5. 在CSS3中唯一引入的伪元素是 ::selection.  
6. 媒体查询，多栏布局  
7. border-image  
**1. 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？**  
**新特性：**1. 拖拽释放(Drag and drop) API  
2. 语义化更好的内容标签（header,nav,footer,aside,article,section）  
3. 音频、视频API(audio,video)  
4. 画布(Canvas) API  
5. 地理(Geolocation) API  
6. 本地离线存储 localStorage 长期存储数据，浏览器关闭后数据不丢失；  
7. sessionStorage 的数据在浏览器关闭后自动删除  
8. 表单控件，calendar、date、time、email、url、search    
9. 新的技术webworker, websocket, Geolocation  
**移除的元素：**  
1. 纯表现的元素：basefont，big，center，font, s，strike，tt，u；  
2. 对可用性产生负面影响的元素：frame，frameset，noframes；  
**支持HTML5新标签：**1. 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，浏览器支持新标签后，还需要添加标签默认的样式（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]-->

**流式布局：所有的元素总是默认的自动向左上角靠近，一般我们只会通过控制元素相对位置的方式使其达到预计的效果，即元素的位置依赖于相邻元素的位置和尺寸。  
  
基于网格的布局：所有的元素可以放在窗体的任意位置，其位置一般是**[**绝对定位**](https://www.baidu.com/s?wd=%E7%BB%9D%E5%AF%B9%E5%AE%9A%E4%BD%8D&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y3uhDznhnYmW99rjbvmWR40ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPHRsnWndPHn3)**的，元素之间的定位没有依赖性。**

**优化网页加载速度**  
1.优化图片资源的格式和大小  
一个网页中，图片资源的大小占比是最多的，而且单个的文件的大小也很可观。因此，在保证图片质量不变的情况下，尽可能的使用高压缩率的图片格式，图片格式可以按照这个优先级选择webp > jpeg > png > bmp。同时也要根据图片展示尺寸来拉取大小最为匹配的图片资源，不要没事就把原图拉下来使用。以前我就遇到过这种情况，一个196\*196大小区域展示的图片，它的文件竟然达到了几兆，最后才发现把1960\*1960分辨率的原图拉下来了。  
2.开启网络压缩  
大部分浏览器在发出请求时，会带上这个标记「Accept-Encoding: gzip, deflate」，表示这个浏览器可以接受以gzip压缩方式传输数据，如果你的网页服务器也支持gzip压缩数据，那么数据以gzip方式传输时，会减少70～80%的流量。  
3.使用浏览器缓存  
同一个站点下面的不同页面，往往都会复用一部分资源文件，如果把这些资源文件设置为可缓存的，那么在刷新或者跳转到另一个页面时，都无须再从网络拉取相关资源，这样就大大加快了网页的加载速度。  
4.减少重定向请求  
有的网站对于不同的终端制作了不同的页面，比如说在手机上访问微博，会从weibo.com重定向至weibo.cn，每一次重定向都会导致浏览器重新发起请求，延长加载时间。对于这种情况，应该尽可能使用响应式设计，一个weibo.com站点覆盖至所有终端。  
5.使用CDN存储静态资源  
CDN是一种静态内容分发网络，它在每个省，甚至每个城市都部署有自己的服务器，用于分发这些静态内容，那么当某个城市的用户要拉取某个资源时，他会首选从本地的CDN服务器上拉取，这样可以保证他最快速的获得该资源。据砖家统计，网络资源中有70%的是静态资源。这就意味着，有70%的内容产生后是不会变化，那么将它们全部放在CDN上面，可以提升这70%的资源的下载速度。  
6.减少DNS查询次数  
很多人喜欢把不同的图片挂在不同当域名下，比如说图片A挂在a.pm-teacher.com，图片B挂在b.pm-teacher.com。当一个网页同时使用图片A和图片B时，浏览器需要查询两个域名，要知道，每次解析域名都是会浪费时间的，所以尽可能的将全部图片放在一个域名下。  
网页加载速度 优化网页 优化网页速度  
7.压缩css和js内容  
这里说的压缩和第2点并不重复，上面提到的压缩是不改变文件内容的压缩。而css和js中有大量的空格和变量命名（如hello="hello word";），如果将这些空格去除，并用简单的字母来代换变量名（如a="hello word";），那么这些css和js原文件的大小也会缩小，这样也对加快拉取速度是有帮助的。  
不知道你有没有看出来，上面提到的优化方案的核心就3点：减少请求数、减少资源大小、找最快的服务器，如果你是一个网站的产品经理，快去找你们的开发确认是否有做过类似的优化吧。  
**跨域、原理、解决方案**

由于浏览器同源策略，凡是发送请求url的协议、域名、端口三者之间任意一与当前页面地址不同即为跨域

JSONP　　这种方式主要是通过动态插入一个script标签。浏览器对script的资源引用没有同源限制，同时资源加载到页面后会立即执行（没有阻塞的情况下）。

 <script>  
       var  
 \_script = document.createElement('script');  
 \_script.type = "text/javascript";  
 \_script.src = "http://localhost:8888/jsonp?callback=f";  
document.head.appendChild(\_script);  
     </script>  
　　缺点：  
　　1、这种方式无法发送post请求（这里 ）  
　　2、另外要确定jsonp的请求是否失败并不容易，大多数框架的实现都是结合超时时间来判定。  
  
　　Proxy代理  
　　这种方式首先将请求发送给后台服务器，通过服务器来发送请求，然后将请求的结果传递给前端。  
  
　　CORS  
　　这是现代浏览器支持跨域资源请求的一种方式。  
  
　　当你使用XMLHttpRequest发送请求时，浏览器发现该请求不符合同源策略，会给该请求加一个请求头：Origin，后台进行一系列处理，如果确定接受请求则在返回结果中加入一个响应头：Access-Control-Allow-Origin;浏览器判断该相应头中是否包含Origin的值，如果有则浏览器会处理响应，我们就可以拿到响应数据，如果不包含浏览器直接驳回，这时我们无法拿到响应数据。

先来介绍一种比较复杂的方法，但这种方法使用起来对所有型号的浏览器都能任意分辨：

**//浏览器类型判定**

function getOs(){

if(navigator.userAgent.indexOf("MSIE")>0) {

return "IE"; //InternetExplor }

else if(isFirefox=navigator.userAgent.indexOf("Firefox")>0){

return "FF"; //firefox

}

else if(isSafari=navigator.userAgent.indexOf("Safari")>0) {

return "SF"; //Safari

}

else if(isCamino=navigator.userAgent.indexOf("Camino")>0){

return "C"; //Camino

}

else if(isMozilla=navigator.userAgent.indexOf("Gecko/")>0){

return "G"; //Gecko

}

else if(isMozilla=navigator.userAgent.indexOf("Opera")>=0){

return "O"; //opera

}else{

return 'Other';

}

}

**//JS实现alert中显示换行的方法:**

function alert\_br(){

var os=getOs();

if(os=='FF' || os=='SF'){ //FireFox、谷歌浏览器用这个

alert('第一行\n第二行');

}else{ //IE系列用这个

alert('第一行\r\n第二行);

}

}

alert\_br();

第二种相对简单，但只能简单地区分IE与其它浏览器：

function alert\_br(){

if(!document.all)//FF/{谷歌浏览器用这个

alert('第一行\n第二行');

}else{ //IE系列用这个

alert('第一行\r\n第二行);

}

}

alert\_br();

**请用JavaScript代码实现从1到10每隔一秒alert**

var count = (function() {  
    var timer;  
    var i = 0;  
    function change(tar) {  
        i++;  
        console.log(i);  
        if (i === tar) {  
            clearTimeout(timer);  
            return false;  
        }  
        timer = setTimeout(function() {  
            change(tar)  
        }, 1000)  
  
    }  
    return change;  
})()  
  
count(50)

**JavaScript数组方法**

**判断是否是数组**

**var *arr*** = [**'l'**,**'o'**,**'c'**,**'a'**,**'l'**,**'S'**,**'t'**,**'o'**,**'r'**,**'a'**,**'g'**,**'e'**];

console.log(***arr* instanceof** Array)  
console.log(Array.isArray(***arr***))

**toString( )：**

数组.toString(); // 把数组变成字符串,去除了[],内容用逗号链接;

例：var arr = ["aaa","bbb","ccc"];

  console.log(arr.toString());//返回 aaa,bbb,ccc

**indexOf / lastIndexOf**

数组.indexOf(元素);   // 给元素，查索引(从前往后)

数组.lastIndexOf(元素); // 给元素，查索引(从后往前)

**Join：**

oin(separator): 将数组的元素组起一个字符串，以separator为分隔符，省略的话则用默认用逗号为分隔符，该方法只接收一个参数：即分隔符。

var arr = [1,2,3];

console.log(arr.join()); // 1,2,3

console.log(arr.join("-")); // 1-2-3

console.log(arr); // [1, 2, 3]（原数组不变）

通过join()方法可以实现重复字符串，只需传入字符串以及重复的次数，就能返回重复后的字符串，函数如下：

function repeatString(str, n) {

return new Array(n + 1).join(str);

}

console.log(repeatString("abc", 3)); // abcabcabc

console.log(repeatString("Hi", 5)); // HiHiHiHiHi

**push()和pop()**

shift()：删除原数组第一项，并返回删除元素的值；如果数组为空则返回undefined 。   
unshift: 将参数添加到原数组开头，并返回数组的长度 。

**4、sort()**

按升序排列数组项——即最小的值位于最前面，最大的值排在最后面。

在排序时，sort()方法会调用每个数组项的 toString()转型方法，然后比较得到的字符串，以确定如何排序。即使数组中的每一项都是数值， sort()方法比较的也是字符串，因此会出现以下的这种情况：

var arr1 = ["a", "d", "c", "b"];

console.log(arr1.sort()); // ["a", "b", "c", "d"]

arr2 = [13, 24, 51, 3];

console.log(arr2.sort()); // [13, 24, 3, 51]

console.log(arr2); // [13, 24, 3, 51](元数组被改变)

为了解决上述问题，sort()方法可以接收一个比较函数作为参数，以便我们指定哪个值位于哪个值的前面。比较函数接收两个参数，如果第一个参数应该位于第二个之前则返回一个负数，如果两个参数相等则返回 0，如果第一个参数应该位于第二个之后则返回一个正数。以下就是一个简单的比较函数：

function compare(value1, value2) {

if (value1 < value2) {

return -1;

} else if (value1 > value2) {

return 1;

} else {

return 0;

}

}

arr2 = [13, 24, 51, 3];

console.log(arr2.sort(compare)); // [3, 13, 24, 51]

如果需要通过比较函数产生降序排序的结果，只要交换比较函数返回的值即可：

function compare(value1, value2) {

if (value1 < value2) {

return 1;

} else if (value1 > value2) {

return -1;

} else {

return 0;

}

}

arr2 = [13, 24, 51, 3];

console.log(arr2.sort(compare)); // [51, 24, 13, 3]

**reverse**()：反转数组项的顺序。

**concat**() ：将参数添加到原数组中。这个方法会先创建当前数组一个副本，然后将接收到的参数添加到这个副本的末尾，最后返回新构建的数组。在没有给 concat()方法传递参数的情况下，它只是复制当前数组并返回副本

var arr = [1,3,5,7];

var arrCopy = arr.concat(9,[11,13]);

console.log(arrCopy); //[1, 3, 5, 7, 9, 11, 13]

console.log(arr); // [1, 3, 5, 7](原数组未被修改)

var arr = [1,3,5,7];

var arrCopy = arr.concat(9,[11,13]);

console.log(arrCopy); //[1, 3, 5, 7, 9, 11, 13]

console.log(arr); // [1, 3, 5, 7](原数组未被修改)

var arrCopy2 = arr.concat([9,[11,13]]);

console.log(arrCopy2); //[1, 3, 5, 7, 9, Array[2]]

console.log(arrCopy2[5]); //[11, 13]

上述代码中，arrCopy2数组的第五项是一个包含两项的数组，也就是说concat方法只能将传入数组中的每一项添加到数组中，如果传入数组中有些项是数组，那么也会把这一数组项当作一项添加到arrCopy2中。

**slice()**

返回从原数组中指定开始下标到结束下标之间的项组成的新数组。slice()方法可以接受一或两个参数，即要返回项的起始和结束位置。在只有一个参数的情况下， slice()方法返回从该参数指定位置开始到当前数组末尾的所有项。如果有两个参数，该方法返回起始和结束位置之间的项——但不包括结束位置的项。

var arr = [1,3,5,7,9,11];

var arrCopy = arr.slice(1);

var arrCopy2 = arr.slice(1,4);

var arrCopy3 = arr.slice(1,-2);

var arrCopy4 = arr.slice(-4,-1);

console.log(arr); //[1, 3, 5, 7, 9, 11](原数组没变)

console.log(arrCopy); //[3, 5, 7, 9, 11]

console.log(arrCopy2); //[3, 5, 7]

console.log(arrCopy3); //[3, 5, 7]

console.log(arrCopy4); //[5, 7, 9]

**splice()**

splice()：很强大的数组方法，它有很多种用法，可以实现删除、插入和替换。

删除：可以删除任意数量的项，只需指定 2 个参数：要删除的第一项的位置和要删除的项数。例如， splice(0,2)会删除数组中的前两项。

插入：可以向指定位置插入任意数量的项，只需提供 3 个参数：起始位置、 0（要删除的项数）和要插入的项。例如，splice(2,0,4,6)会从当前数组的位置 2 开始插入4和6。  
替换：可以向指定位置插入任意数量的项，且同时删除任意数量的项，只需指定 3 个参数：起始位置、要删除的项数和要插入的任意数量的项。插入的项数不必与删除的项数相等。例如，splice (2,1,4,6)会删除当前数组位置 2 的项，然后再从位置 2 开始插入4和6。

splice()方法始终都会返回一个数组，该数组中包含从原始数组中删除的项，如果没有删除任何项，则返回一个空数组。

var arr = [1,3,5,7,9,11];

var arrRemoved = arr.splice(0,2);

console.log(arr); //[5, 7, 9, 11]

console.log(arrRemoved); //[1, 3]

var arrRemoved2 = arr.splice(2,0,4,6);

console.log(arr); // [5, 7, 4, 6, 9, 11]

console.log(arrRemoved2); // []

var arrRemoved3 = arr.splice(1,1,2,4);

console.log(arr); // [5, 2, 4, 4, 6, 9, 11]

console.log(arrRemoved3); //[7]

**every()**

every()：判断数组中每一项都是否满足条件，只有所有项都满足条件，才会返回true。

**var *arr2*** = [111,222,333,444,555];  
*/\*arr2.every(function(element,index,array){  
 console.log(element)  
 console.log(index)  
 console.log(array)  
 console.log('-----')  
 return true  
})\*/*console.log(  
 ***arr2***.every(**function**(element,index,array){  
 **if**(element>200){  
 **return true** }  
 **return false** })  
)

**fliter:**

// 对数组中每一项运行回调函数，该函数返回结果是true的项组成的新数组

// 新数组是有老数组中的元素组成的，return为ture的项;

**var *arr2*** = [111,222,333,444,555];

console.log(***arr2***.filter(**function**(element, index, array){  
 **if**(element % 2 === 0){*//只要是奇数，就组成数组;(数组中辨别元素)* **return true** }**else** {  
 **return false** }  
}))

**详解 Cookie、 LocalStorage 与 SessionStorage**

**Cookie**

Cookie 是小甜饼的意思。顾名思义，cookie 确实非常小，它的大小限制为4KB左右。它的主要用途有保存登录信息，比如你登录某个网站市场可以看到“记住密码”，这通常就是通过在 Cookie 中存入一段辨别用户身份的数据来实现的。

**localStorage**

localStorage 是 HTML5 标准中新加入的技术，它并不是什么划时代的新东西。早在 IE 6 时代，就有一个叫 userData 的东西用于本地存储，而当时考虑到浏览器兼容性，更通用的方案是使用 Flash。而如今，localStorage 被大多数浏览器所支持，如果你的网站需要支持 IE6+，那以 userData 作为你的 polyfill 的方案是种不错的选择。

**sessionStorage**

sessionStorage 与 localStorage 的接口类似，但保存数据的生命周期与 localStorage 不同。做过后端开发的同学应该知道 Session 这个词的意思，直译过来是“会话”。而 sessionStorage 是一个前端的概念，它只是可以将一部分数据在当前会话中保存下来，刷新页面数据依旧存在。但当页面关闭后，sessionStorage 中的数据就会被清空。

**三者的异同**

| **特性** | **Cookie** | **localStorage** | **sessionStorage** |
| --- | --- | --- | --- |
| 数据的生命期 | 一般由服务器生成，可设置失效时间。如果在浏览器端生成Cookie，默认是关闭浏览器后失效 | 除非被清除，否则永久保存 | 仅在当前会话下有效，关闭页面或浏览器后被清除 |
| 存放数据大小 | 4K左右 | 一般为5MB | |
| 与服务器端通信 | 每次都会携带在HTTP头中，如果使用cookie保存过多数据会带来性能问题 | 仅在客户端（即浏览器）中保存，不参与和服务器的通信 | |
| 易用性 | 需要程序员自己封装，源生的Cookie接口不友好 | 源生接口可以接受，亦可再次封装来对Object和Array有更好的支持 | |

**图片预加载和懒加载**

预加载：

预加载就是通过[Image](https://developer.mozilla.org/zh-CN/docs/Web/API/HTMLImageElement/Image)对象，给这个对象添加“src”属性，并可以将此对象缓存起来，以后再使用。

<img src="blank.png" data-src="img/p2/1.png" class="img1">

我先给img赋个空白图，然后通过JS来给src赋data-src中的值，预加载的逻辑从网上找到了相关代码。

function loadImage(url, callback) {

var img = new Image();

img.src = url;

if (img.complete) { 如果图片已经存在于浏览器缓存，直接调用回调函数防止IE6不执行

callback.call(img);

return;

}

img.onload = function () {

callback.call(img);//将回调函数的this替换为Image对象

};

};

懒加载：

**二、懒加载**

一般懒加载就是当你做滚动到页面某个位置，然后再显示当前位置的图片，这样做可以减少页面请求。

参考了两个比较火的开源库，[jquery\_lazyload](https://github.com/tuupola/jquery_lazyload)和[layzr.js](https://github.com/callmecavs/layzr.js)，两者的人气都很足，不过前者是需要引入jQuery库的，后者不需要。

当页面过长的时候就会出现滚动条，而当年滚动到下面的时候，上面的页面就会看不到，下图中绿色部分就是那隐藏的页面。

通过计算，可以获取图片的两个距离值，图中标注了，然后判断是否在当前可视区域的顶部和底部的范围内，如果是就加载图，不是就不加载。

假设滚动条是在body中，那么当前可视区域的范围是：

viewTop = window.scrollY || window.pageYOffset;

viewBottom = window.innerHeight + viewTop;

而图片的top和height是：

nodeTop = node.getBoundingClientRect().top + viewTop;

nodeBottom = nodeTop + node.offsetHeight;

在上面出现了几个尺寸的概念。

[Window.scrollY](https://developer.mozilla.org/zh-CN/docs/Web/API/Window/scrollY) 与 [Window.pageYOffset](https://developer.mozilla.org/en-US/docs/Web/API/Window/pageYOffset)意思差不多，指的是滚动条顶部到网页顶部的距离。

[Window.innerHeight](https://developer.mozilla.org/zh-CN/docs/Web/API/Window/innerHeight) 表示该容器中页面视图区的高度（减去边框宽度）。

[Element.getBoundingClientRect()](https://developer.mozilla.org/zh-CN/docs/Web/API/Element/getBoundingClientRect) 方法返回元素的大小及其相对于视口的位置。

[HTMLElement.offsetHeight](https://developer.mozilla.org/zh-CN/docs/Web/API/HTMLElement/offsetHeight) = height+padding+border。

更多的尺寸属性可以参考《[JavaScript中尺寸、坐标](http://www.cnblogs.com/strick/p/4826273.html)》

最后在给某个容器绑定“scroll”事件，上面的话是给“window”绑定。

页可见区域宽： document.body.clientWidth;

网页可见区域高： document.body.clientHeight;

网页可见区域宽： document.body.offsetWidth (包括边线的宽);

网页可见区域高： document.body.offsetHeight (包括边线的宽);

网页正文全文宽： document.body.scrollWidth;

网页正文全文高： document.body.scrollHeight;

网页被卷去的高： document.body.scrollTop;

网页被卷去的左： document.body.scrollLeft;

网页正文部分上： window.screenTop;

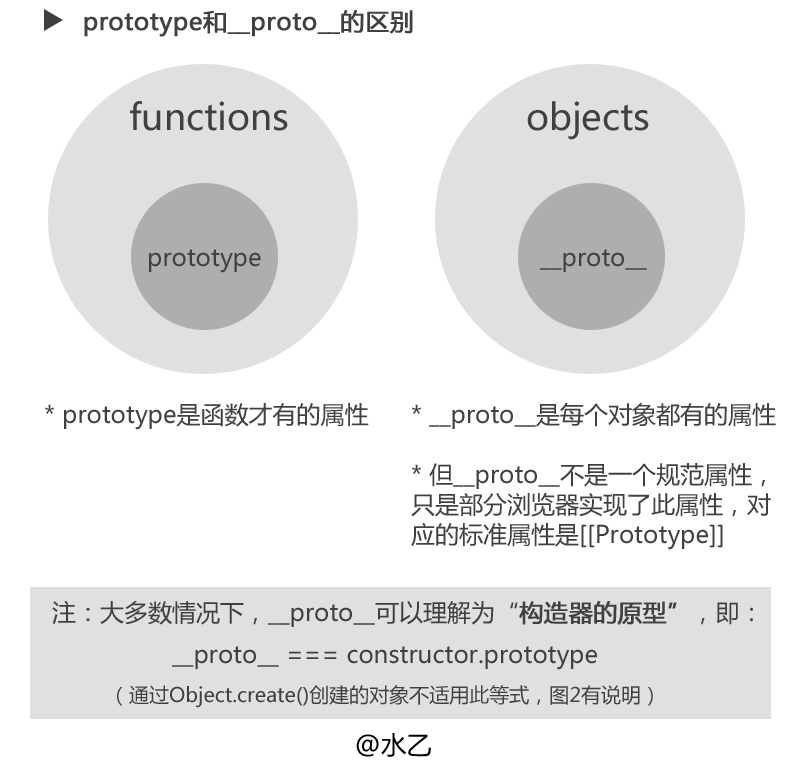
网页正文部分左： window.screenLeft;

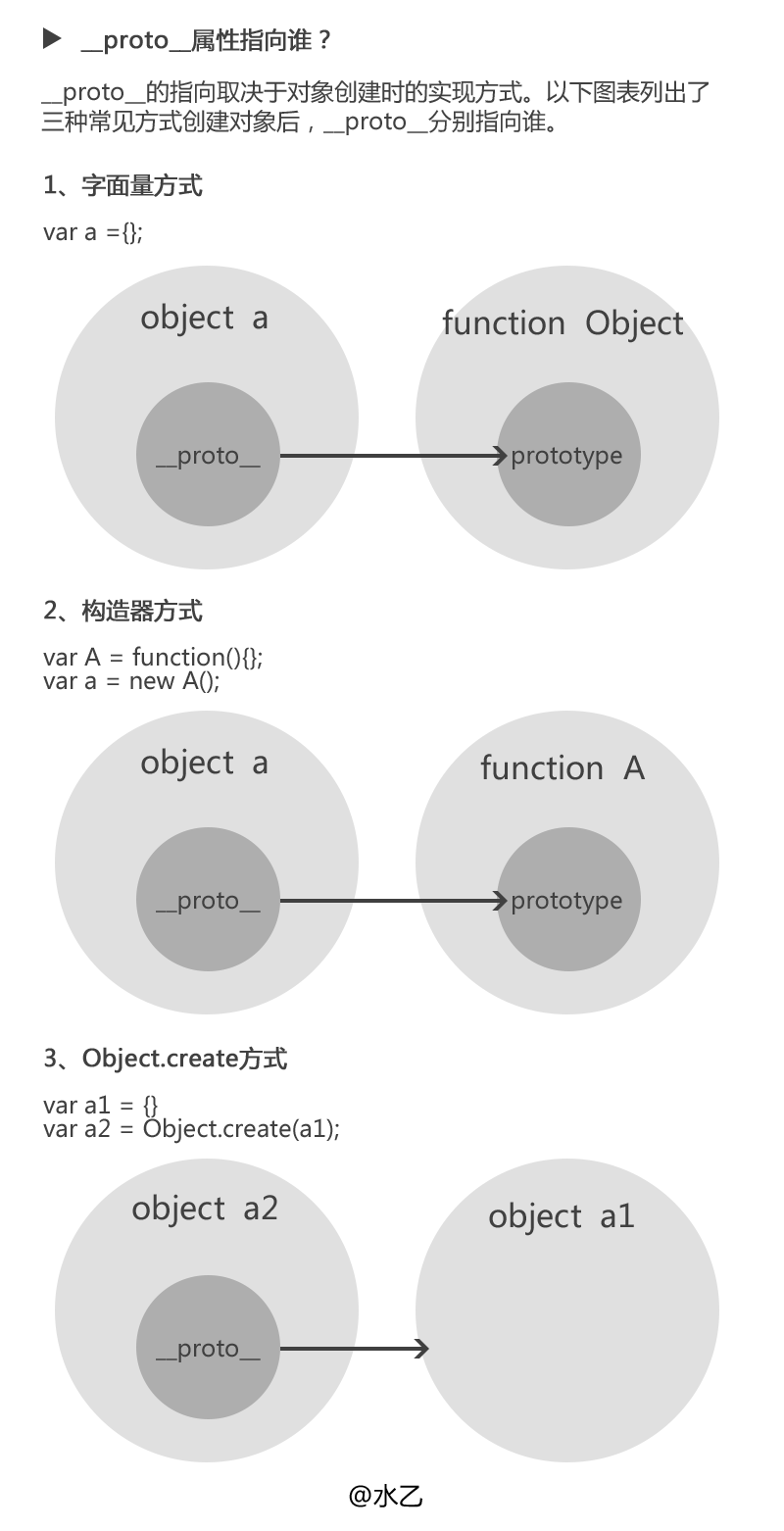
屏幕分辨率的高： window.screen.height;

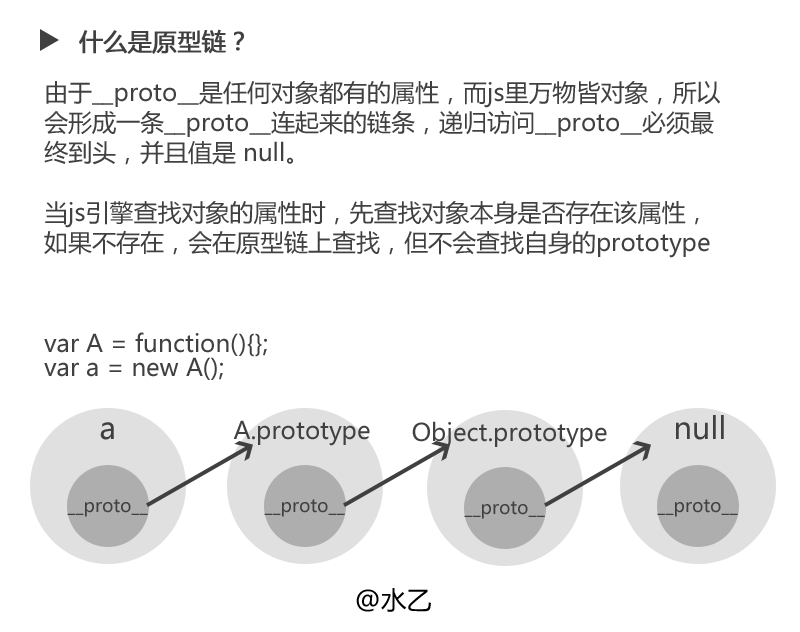
屏幕分辨率的宽： window.screen.width;

屏幕可用工作区高度： window.screen.availHeight;

1、原型-原型链：







**原型链问题**

**1.什么是prototype，constructor，\_\_proto\_\_**   
prototype是构造函数的属性，指的就是构造函数的原型，在生成实例的时候，js会根据构造函数的prototype属性将该属性下的对象生成为父类,  
注意，只有构造函数这个属性才有这种效果哦~如果一个构造函数没有指定该属性，那么该属性下的\_\_proto\_\_会默认的指向原生Object的原型对象,该属性会变成一个对象，其中constructor属性指向本身。  
​constructor如果通俗的理解，可以理解成原型对象的构造函数，当把一个对象（对象字面量）赋给一个构造函数的原型，constructor 会被复写，如果没有进行prototype的操作的话，constructor是函数声明时指定的，指定为本身：  
  
例如：  
function A() {}　　  
        则 A.prototype.constructor === A; 但是这个属性往往是不准确的：  
        function A() {}  
        function B() {}  
        B.prototype = new A();  
​        var b = new B();  
        b.constructor; // A  
 上面的代码，按照constructor的含义，b的constructor应该指向B，但是确指向了A，原因如下  
 A没有进行prototype操作，所以其constructor指向本身；　B.prototype = new A();

之后B.prototype.constructor === A;         B.prototype被复写为A的实例，B.prototype.constructor === A;

而b是B的实例则b.constructor === A；  
如果想要实现继承，一般要进行constructor修复，即：　　B.prototype = new A();　　B.prototype.constructor = B;  
\_\_proto\_\_  
​可以这么说，js的原型链就是通过这个属性串联起来的，\_\_proto\_\_属性指向他的父类，在调用一个对象的属性或者方法的时候就是通过\_\_proto\_\_这一属性指向的对象一层一层的向上查找的。上面的一句：在生成实例的时候，js会根据构造函数的prototype属性将该属性下的对象生成为父类,在这里可以改为，在生成实例的时候，js会根据构造函数的prototype属性将该属性下的对象引用到实例的\_\_proto\_\_属性下。  
​**2.什么是原型链**  
​我们首先来说说继承，继承是面向对象语言的重要机制，通俗地讲就是子类可以拥有父类的方法和属性，js的原型链实际上也是一种继承，在ECMAScript标准中，只支持实现继承，而其实现实现继承就是主要依靠于原型链实现的。

那么，我们再来说说原型，原型其实就是上述所说的继承中的父类。

这样，原型链 显而易见的 可以理解为，利用原型串起一个继承链，让一个引用类型继承另一个引用类型的属性和方法，再以此类推下去.  
**3.​构造函数，原型，与实例的关系：**​这三者的关系用一句话概括为，每个构造函数都有一个原型，当new 一个构造函数的时候就会生成一个原型链上携带该构造函数的原型的实例。  
**4.原型链的生成过程：**  
function parent(){   
     this.a = "world";   
     this.b = "world";   
}   
  
parent.prototype = {   
   a : "aaa",   
   b : "bbb",   
   c : "!"   
  }  
 function temp(){  
  this.a = "hello";  
 }  
temp.prototype = new parent();  
var child = new temp();  
console.log(child.a +" " + child.b+child.c);//hello world!​​​​​​  
上面的代码运行结果就为 ： “hello world！”生成的原型链即是  
child(temp的实例) > \_\_proto\_\_ > temp.prototype(parent的实例) > \_\_proto\_ >parent.prototype > \_\_proto\_\_ > Object.prototype >\_\_proto\_\_ > null  
​**5.常用函数**  
function extend(child, parent) {​  
      function temp() {  
          this.constructor = child；  
​      }  
     temp.prototype = parent.prototype;  
     child.prototype = new temp();  
​}  
extend函数 两个参数都为构造函数，目的是让一个构造函数（子类）继承另一个构造函数（父类）的原型，并且不调用父类的构造函数（有时候构造函数会需要一些参数或者做一些事情破坏了原型链），这个方法可以用来生成想要的原型链哦。​