1. **列举几种后端通讯的方法，分别使用的场景。**

跨域请求存在的原因：由于浏览器的同源策略，即属于不同域的页面之间不能相互访问各自的页面内容。

**跨域的场景**：

1.域名不同 www.yangwei.com 和www.wuyu.com 即为不同的域名）

2.二级域名相同，子域名不同（www.wuhan.yangwei.com www.shenzheng.yangwei.com 为子域不同）

3.端口不同，协议不同  （ http://www.yangwei.com 和https://www.yangwei.com属于跨域www.yangwei.con:8888和www.yangwei.con:8080)

**跨域的方式：（内容较多，需掌握CORS和jsonp，其他内容也要了解）**

**1.前端的方式**: possMessage，window.name,document.domain,image.src(得不到数据返回)，jsonP(script.src后台不配合得不到数据返回)，style.href（得不到数据返回)

一.imge.src,script.src,style.href 不受同源策略的影响可以加载其他域的资源，可以用这个特性，向服务器发送数据。最常用的就是使用image.src 向服务器发送前端的错误信息。image.src 和style.href 是无法获取服务器的数据返回的，script.src 服务器端配合可以得到数据返回。

二possMessage,window.name,document.domain 是两个窗口直接相互传递数据。

（1）possMessage 是HTML5中新增的，使用限制是 必须获得窗口的window 引用。IE8+支持，firefox，chrome,safair,opera支持

  （2）window.name ，在一个页面中打开另一个页面时，window.name 是共享的，所以可以通过window.name 来传递数据，window.name的限制大小是2M，这个所有浏览器都支持,且没有什么限制。

3） document.domain 将两个页面的document.domain 设置成相同，document.domain 只能设置成父级域名，既可以访问，使用限制：这顶级域名必须相同

**2.纯后端方式: CORS，服务器代理**

CORS 是w3c标准的方式，通过在web服务器端设置：响应头Access-Cntrol-Alow-Origin 来指定哪些域可以访问本域的数据，ie8&9(XDomainRequest),10+,chrom4 ，firefox3.5,safair4，opera12支持这种方式。

服务器代理，同源策略只存在浏览器端，通过服务器转发请求可以达到跨域请求的目的，劣势：增加服务器的负担，且访问速度慢。

**3.前后端结合:JsonP**

script.src 不受同源策略的限制，所以可以动态的创建script标签，将要请求数据的域写在src 中参数中附带回调的方法，服务器端返回回调函数的字符串，并带参数。

如 script.src="http://www.yangwei.com/?id=001&callback=getInfoCallback,服务器端返回 getInfoCallBack("name:yangwei;age:18") 这段代码会直接执行，在前面定义好getInfoCallBack函数，既可以获得数据并解析。 这种是最长见的方式。

**4.webSocke（了解性拓展）**

**[服务端推送websocket和sse场景及应用](https://www.iflym.com/index.php/java-network/201607040001.html" \o "Permanent Link to 服务端推送websocket和sse场景及应用)**

**应用场景**

都可以进行服务端推送,并且都是使用长连接来进行.但两者的实现又有一点不同,sse仍使用http协议,并且使用相同的链接发送正常的http协议报文.而websocket是使用http协议进行握手,然后再使用同一个链接进行websocket协议的通信.

**websocket可以进行双向的通信,**即服务端可以往客户端发信息,客户端也可以向服务端发信息.而sse是单向的,只能由服务端往客户端发.

**websocket自带连接的保持**,即通过ping/pong协议保证连接可以始终维持,sse没有这个保证,不过可以参考ping/pong协议,自己周期性地发送信息来同样地进行处理.比如,5秒往客户端发一个特别的信息(通过type/name进行区分).其次,因为是基于浏览器的使用,sse有一个特性,就是浏览器发现一个连接断掉了,就会自动地进行重联,即重新发送请求.这样,服务端也不用担心连接被断开,不过需要处理新的请求必须和上一次请求的内容相连续,以及新的推送注册.

因为都是使用http协议进行起始处理,因此在签权上都可以使用到http协议本身的一些东西,比如header/cookie签权.在相应的握手阶段,通过读取cookie(session)来保证相应的请求必须是经过授权的,也可以用于定位使用人.甚至可以通过这些信息保证单个用户只能有一个请求,避免重复请求

由于都是基于浏览器使用,因此建议的数据传输都是文本型.虽然websocket支持二进制frame传输,不过一些都不建议使用.sse只能传输文本

不管是websocket还是sse,在用于通信时,都建议只用于进行数据的推送,而不是进行完整的应用处理.这里可以理解为,常规的业务处理仍然交给后端的服务来处理.这样,即可以使用之前的业务开发的优势,又可以使用推送的优势.而不是走向另一个级端,即所有的信息都想通过推送来传递.

**开发方式**

websocket开发首选netty,因为netty对协议的封装已经做到了完全的支持.通过 HttpServerCodec作为握手协议,WebSocketServerProtocolHandler作为协议处理,然后再加一个自己的handler,就完成了相应的业务处理.同时在性能上,netty在一个ws的请求建立起来之后,会自动地去除httpServerCodec相关的handler,这样保证后续的处理都是按照ws的协议来进行.

sse开发首选jersey,jersey-media-sse提供了相应的sse支持,并且通过与rest相集成,开发一个sse就跟普通的业务开发相同.

ws和sse在文本支持上都只支持utf-8编码,因此在处理上需要注册编码方式.同时在使用sse时,如果后端第一次进行响应时,相应的编码不对.chrome会直接报错,包括utf8都会报错(这是之前后端开发的一个问题),可以修正或者增加相应的拦截器,保证后端content-type响应中的charset=UTF-8.

ws和sse都可以通过nginx进行代理转发.ws的处理只需要设置http版本,以及重新转发前端的Upgrade和Connection头即可.而sse,也可以通过禁用buffer来处理.参考 http://stackoverflow.com/questions/27898622/server-sent-events-stopped-work-after-enabling-ssl-on-proxy

**特定实现**

为保证在开发时推送类的和业务类的系统不会耦合在一起,或者同一个应用内有两种处理模式的功能存在.建议直接在系统层就开发2个不同的系统,一个专门用于推送,另一个用于相应的业务处理.然后业务处理后的数据,需要再交由推送处理,则可以在后端进行通过消息系统进行中转,如kafka(持久保证)或redis(内存订阅)等

因为二者在ie上的支持都很有限,因此不建议在ie上进行尝试

使用sse还是websocket,取决于是否需要前台交互,还取决于对后端的支持技术的了解程序.比如,了解jersey多一点,还是netty多一点.由于最近netty进行微服务化底层通信支持越来越流行,个人更倾向于使用websocket.但如果仅仅是一个简单的推送功能,又不希望修改代码,那也可以使用jersey(毕竟之前的系统就是在上面进行开发的)

需要后端有的时候需要进行定向发送或者是群发,这种需求ws和sse的实现中都有相应的处理.如ChannelGroup和SseBroadcaster,这样在后端获取到一个消息,需要进行路由时就可以从这里面拿相应的channel信息.不过,前提是对各个channel上进行了特定的消息绑定,这样就好区分具体的路由信息.具体路由策略可以在建立时绑定session,后续通过session来路由.

1. **设计一个幻灯应用，需要列举选择的基础框架、项目的基础框**

**架和代码管理、幻灯数据的存储和读取，部分特效的实现，可以只写思路，后续面聊。**

本题并没有找到好的答案解析，自己也没有好的思路，有比较好想法的同学可以分享一下。

1. **请写出至少20个HTML5标签**

<article> <aside> <audio> <canvas> <datalist> <command>  <details> <embed>

<figcaption> <figure> <footer> <header> <hgroup> <keygen> <mark> <nav>

<section> <time> <video> <summary>

1. **列举5种IE hastlayout的属性及其值**

**haslayout** 是Windows Internet Explorer渲染引擎的一个内部组成部分。在Internet Explorer中，一个元素要么自己对自身的内容进行计算大小和组织，要么依赖于父元素来计算尺寸和组织内容。为了调节这两个不同的概念，渲染引擎采用 了 hasLayout 的属性，属性值可以为true或false。当一个元素的 hasLayout 属性值为true时，我们说这个元素有一个布局（layout）

部分的 IE 显示的错误，都可以通过激发元素的 haslayout 属性来修正。可以通过设置 css 尺寸属性(width/height)等来激发元素的 haslayout，使其“拥有布局”。如下所示，通过设置以下 css 属性即可。

\* display: inline-block  
\* height: (任何值除了auto)  
\* float: (left 或 right)  
\* position: absolute  
\* width: (任何值除了auto)  
\* writing-mode: tb-rl  
\* zoom: (除 normal 外任意值)

Internet Explorer 7 还有一些额外的属性(不完全列表):

\* min-height: (任意值)  
\* max-height: (除 none 外任意值)  
\* min-width: (任意值)  
\* max-width: (除 none 外任意值)  
\* overflow: (除 visible 外任意值)  
\* overflow-x: (除 visible 外任意值)  
\* overflow-y: (除 visible 外任意值)  
\* position: fixed

1. **简述jpg。Gif。png-8.png-24的区别，分别使用场景**

gif、jpg、png格式的图片在网站制作中的区别

**Gif格式特点:**  
　　1.透明性,Gif是一种布尔透明类型，既它可以是全透明，也可以是全不透明，但是它并没有半透明（alpha透明）。  
　　2.动画,Gif这种格式支持动画。  
　　3.无损耗性,Gif是一种无损耗的图像格式，这也意味着你可以对gif图片做任何操作也不会使得图像质量产生损耗。  
　　4.水平扫描,Gif是使用了一种叫作LZW的算法进行压缩的，当压缩gif的过程中，像素是由上到下水平压缩的，这也意味着同等条件下，横向的gif图片比竖向的gif图片更加小。例如500\*10的图片比10\*500的图片更加小  
　　5.间隔渐进显示,Gif支持可选择性的间隔渐进显示  
　　由以上特点看出只有256种颜色的gif图片不适合照片，但它适合对颜色要求不高的图形（比如说图标，图表等），它并不是最优的选择，我们会在后面中看到png是最优的选择。  
　　**Jpeg格式特点:**  
　　1.透明性,它并不支持透明。  
　　2.动画,它也不支持动画。  
　　3.损耗性,除了一些比如说旋转（仅仅是90、180、270度旋转），裁切，从标准类型到先进类型，编辑图片的原数据之外，所有其它操作对jpeg图像的处理都会使得它的质量损失。所以我们在编辑过程一般用png作为过渡格式。  
　　4.隔行渐进显示,它支持隔行渐进显示（但是ie浏览器并不支持这个属性，但是ie会在整个图像信息完全到达的时候显示）。  
　　由上可以看出Jpeg是最适web上面的摄影图片和数字照相机中。  
**Png格式特点:**  
　　1.类型,Png这种图片格式包括了许多子类，但是在实践中大致可以分为256色的png和全色的png，你完成可以用256色的png代替gif，用全色的png代替jpeg  
　　2.透明性,Png是完全支持alpha透明的（透明，半透明，不透明），尽管有两个怪异的现象在ie6（下面详细讨论）  
　　3.动画,它不支持动画  
　　PNG图片格式现在包含三种类型：  
　　1.PNG8256色PNG的别名  
　　2.PNG24全色PNG的别名  
　　3.PNG32全色PNG的别名  
　　基本上PNG32就是PNG24，但是附带了全alpha通道。就是说每个像素上不仅存储了24位真色彩信息还存储了8位的alpha通道信息，就如同GIF能存储透明和不透明信息一样。当我们把图片放到不太搭配的背景上的时候，透明PNG图片的边缘会显示得更加平滑。  
　　当然，我也知道你的想法，“但是Photoshop也能生成带透明通道的PNG图片！”我也知道，它只是表面上这么说是PNG24，让我也产生困惑了。  
　　作为一个伤感的Fireworks倡导者，我只使用PNG32支持附带alpha通道的真色彩图片。不管怎样，如果你习惯使用Photoshop，你就应该知道，Photoshop在“存储为WEB格式”中只提供PNG8和PNG24两种PNG格式。  
　　我敢肯定你经常会勾选“支持透明”选项，以获得带有透明度的PNG图片，但是这样你就获取了一张PNG32图片。——Photoshop只是觉得把PNG32这个名称给隐藏掉了。奇怪吧？……  
**对png8的误解**  
　　Png8的在ie中的怪异表现：  
　　半透明的png8在ie6以下的浏览器显示为全透明。  
　　Alpha透明的全色PNG（png32）在ie6中会出现背景颜色（通常是灰色）。  
　　由上面可以总结：  
　　（a）全透明的png8可以在任一浏览器正常显示（就像gif一样）。半透明的png8在除了ie6及其以下的浏览器下错误的显示成全透明，其它浏览器都能正常显示半透明。这个bug并不需要特殊对待，因为在不支持半透明的浏览器下只是显示为全透明，对用户体验影响不大，它反而是透明gif的加强版。  
　　（b）第二个bug没有什么好的方法解决，只能通过影响性能的方法AlphaImageLoader与需要加特殊标签（VML）。  
　　因此得出结论就是：请使用PNG8。  
　　Png8的软件问题：  
　　Photoshop只能导出布尔透明的PNG8。  
　　Fireworks既能导出布尔透明的PNG8，也能导出alpha透明的PNG8.

1. **算法题：**

**有一个长度为n-1的数组，包含1-n中不重复的乱序的数，求寻找范围内不在数组中的数，考虑空间占用，性能优化，溢出等情况，至少写两个算法**

 当n不太大时，可以考虑求和。先算出1~n的所有数的和，然后减去数组中出现的所有自然数的和。时间复杂度为O(n)，空间复杂度O(1)。这种方法的缺点是n不能太大，n比较大时，求和容易溢出。

用位图。从头到尾的扫描整个数组，把出现的数相应的位设置为1.然后再扫描位图，找出不为1的那一位，即为要找的数。这种方法的时间复杂度为O(n),空间复杂度为O(n)。

异或有个很巧妙的地方：同一变量和该变量与另一变量的异或值的异或等于这个变量自身。所以我们可以把1~n的所有数异或，再把数组中出现的所有数异或，然后再把这两个异或的结果异或，最后得到的值即为我们要找的值。这样时间复杂度为O(n),空间复杂度为O（1）。在空间上比第二种方法要好，而且不会出现第一种方法中所说的溢出问题。

1. **实现以下方法（与标准一致）**

**Element.prototype .getElemantsByclassname**

element.prototype.getElementsByClassName = function (searchClass, node,tag) {

  if(document.getElementsByClassName){

 var nodes =  (node || document).getElementsByClassName(searchClass),result = [];

for(var i=0 ;node = nodes[i++];){

        if(tag !== "\*" && node.tagName === tag.toUpperCase()){

          result.push(node)

        }

      }

      return result

    }else{

      node = node || document;

      tag = tag || "\*";

      var classes = searchClass.split(" "),

      elements = (tag === "\*" && node.all)? node.all : node.getElementsByTagName(tag),

      patterns = [],

      current,

      match;

      var i = classes.length;

      while(--i >= 0){

        patterns.push(new RegExp("(^|\\s)" + classes[i] + "(\\s|$)"));

      }

      var j = elements.length;

      while(--j >= 0){

        current = elements[j];

        match = false;

        for(var k=0, kl=patterns.length; k<kl; k++){

          match = patterns[k].test(current.className);

          if (!match)  break;

        }

        if (match)  result.push(current);

      }

      return result;

    }

  }

**Function.Prototype.bind**

Function.prototype.bind = function (oThis) {

if (typeof this !== "function") {

throw new TypeError("bind function error");

}

var aArgs = Array.prototype.slice.call(arguments,1),

fToBind = this,

fBound = function () {

return fToBind.apply(oThis || window,aArgs.concat(Array.prototype.slice.call(arguments)));

};

return fBound;

};

1. **能够设置文本加粗的样式属性是什么**

字体加粗（font-weight）

功能：用于设置字体笔划的粗细。

属性值：正常度 - normal

相对度 - bold, bolder, light, lighter

渐变度 - 100, 200, 300, 400(相当于normal), 500, 600, 700(相当于 bold、 lighter、 bolder、以及数值100－900。

语法为：h1 {font-weight: 属性值}

1. **编写一个方法去掉一个数组的重复元素**

**1.遍历数组法**

最简单的去重方法，**实现思路：**新建一新数组，遍历传入数组，**值不在新数组就加入该新数组中**；**注意点**：判断值是否在数组的方法“indexOf”是ECMAScript5 方法，IE8以下不支持，需多写一些兼容低版本浏览器代码，源码如下：

// 最简单数组去重法

function unique1(array){

var n = []; //一个新的临时数组

//遍历当前数组

for(var i = 0; i < array.length; i++){

//如果当前数组的第i已经保存进了临时数组，那么跳过，

//否则把当前项push到临时数组里面

if (n.indexOf(array[i]) == -1) n.push(array[i]);

}

return n;

}

// 判断浏览器是否支持indexOf ，indexOf 为ecmaScript5新方法 IE8以下（包括IE8， IE8只支持部分ecma5）不支持

if (!Array.prototype.indexOf){

// 新增indexOf方法

Array.prototype.indexOf = function(item){

var result = -1, a\_item = null;

if (this.length == 0){

return result;

}

for(var i = 0, len = this.length; i < len; i++){

a\_item = this[i];

if (a\_item === item){

result = i;

break;

}

}

return result;

}

}

**2.对象键值对法**

该方法执行的速度比其他任何方法都快， 就是占用的内存大一些；**实现思路：**新建一js对象以及新数组，遍历传入数组时，**判断值是否为js对象的键，**不是的话给对象新增该键并放入新数组。**注意点：**判断是否为js对象键时，会自动对传入的键执行“toString()”，不同的键可能会被误认为一样；例如： a[1]、a["1"] 。解决上述问题还是得调用“indexOf”。

// 速度最快， 占空间最多（空间换时间）

function unique2(array){

var n = {}, r = [], len = array.length, val, type;

for (var i = 0; i < array.length; i++) {

val = array[i];

type = typeof val;

if (!n[val]) {

n[val] = [type];

r.push(val);

} else if (n[val].indexOf(type) < 0) {

n[val].push(type);

r.push(val);

}

}

return r;

}

**3.数组下标判断法**

还是得调用“indexOf”性能跟方法1差不多，**实现思路**：如果当前数组的第i项在当前数组中第一次出现的位置不是i，那么表示第i项是重复的，忽略掉。否则存入结果数组。

function unique3(array){

var n = [array[0]]; //结果数组

//从第二项开始遍历

for(var i = 1; i < array.length; i++) {

//如果当前数组的第i项在当前数组中第一次出现的位置不是i，

//那么表示第i项是重复的，忽略掉。否则存入结果数组

if (array.indexOf(array[i]) == i) n.push(array[i]);

}

return n;

}

**4.排序后相邻去除法**

虽然原生数组的”sort”方法排序结果不怎么靠谱，但在不注重顺序的去重里该缺点毫无影响。**实现思路：**给传入数组排序，排序后相同值相邻，然后遍历时新数组只加入不与前一值重复的值。

// 将相同的值相邻，然后遍历去除重复值

function unique4(array){

array.sort();

var re=[array[0]];

for(var i = 1; i < array.length; i++){

if( array[i] !== re[re.length-1])

{

re.push(array[i]);

}

}

return re;

}

**5.优化遍历数组法**

**实现思路**：获取没重复的最右一值放入新数组。（检测到有重复值时终止当前循环同时进入顶层循环的下一轮判断）

// 思路：获取没重复的最右一值放入新数组

function unique5(array){

var r = [];

for(var i = 0, l = array.length; i < l; i++) {

for(var j = i + 1; j < l; j++)

if (array[i] === array[j]) j = ++i;

r.push(array[i]);

}

return r;

}

1. **编写一个布局，页面宽度自适应，最小宽度300px，左边定宽35%，右边定宽65%**

<div class="container">

<div class="left"></div>

<div class="right"></div>

</div>

<style>

.container{

height: 600px;

\_width: 300px;

min-width: 300px;

}

.left{

width: 35%;

height: 100%;

background: #ff0;

float: left;

}

.right{

overflow:hidden;

width: 65%;

height: 100%;

background: #0f0;

}

</style>

1. **谈谈对html5的了解**

1.良好的移动性，以移动设备为主。

2.响应式设计，以适应自动变化的屏幕尺寸

3.支持离线缓存技术，webStorage本地缓存

4.新增canvas，video，audio等新标签元素。新特殊内容元素：article，footer，header，nav，section等，新的表单控件：calendar，date，time，email，url，search。

5.地理定位...

6.新增webSocket/webWork技术

1. **Js面向对象的几种方式**

1.对象的字面量 var obj = {}

2.创建实例对象 var obj = new Object();

3.构造函数模式 function fn(){} , new fn();

4.工厂模式：用一个函数，通过传递参数返回对象。function fn(params){var obj =new Object();obj.params = params; return obj;},fn(params);

5.原型模式：function clock(hour){} fn.prototype.hour = 0; new clock();

首先，每个函数都有一个prototype(原型)属性，这个指针指向的就是clock.prototype对象。而这个原型对象在默认的时候有一个属性constructor，指向clock，这个属性可读可写。而当我们在实例化一个对象的时候，实例newClock除了具有构造函数定义的属性和方法外（注意，只是构造函数中的）,还有一个指向构造函数的原型的指针，ECMAScript管他叫[[prototype]]，这样实例化对象的时候，原型对象的方法并没有在某个具体的实例中，因为原型没有被实例。

1. **在css中哪个属性会影响dom读取文档流的顺序**

1.direction, writing-mode

1. **前端页面有哪三层构成，分别是什么，作用是什么**

Css: 层叠样式表 ,表现, 由css负责创建。css对“如何显示有关内容”的问题做出了回答。

Html:超文本标记语言 ,结构, 由 HTML 或 xhtml之类的标记语言负责创建。标签，也就是那些出现在尖括号里的单词，对网页内容的语义含义做出了描述，但这些标签不包含任何关于如何显示有关内容的信息。例如，P标签表达了这样一种语义：“这是一个文本段。”

Js: 客户端脚本语言 ,行为, 内容应该如何对事件做出反应

1. **Css的基本语句构成是？**

语法:  
（自定义的样式名称）｛  
 样式内容  
｝

1. **如何对网站的文件和资源进行优化**

1．文件合并（目的是减少http请求）

2．文件压缩 （目的是直接减少文件下载的体积）

3.使用cdn托管资源

4.使用缓存

5.gizp压缩你的js和css文件

6.meta标签优化（title,description,keywords）,heading标签的优化,alt优化

7.反向链接，网站外链接优化

1. **Javascipt的本地对象，内地对象和宿主对象**

本地对象：Object、Function、Array、String、Boolean、Number、Date、RegExp、Error、EvalError、RangeError、ReferenceError、SyntaxError、TypeError、URIError, 简单来说，本地对象就是 ECMA-262 定义的类.

**内置对象：**ECMA-262 把内置对象（built-in object）定义为“由 ECMAScript 实现提供的、独立于宿主环境的所有对象，在 ECMAScript 程序开始执行时出现”。这意味着开发者不必明确实例化内置对象，它已被实例化了。

同样是“独立于宿主环境”。根据定义我们似乎很难分清“内置对象”与“本地对象”的区别。而ECMA-262 只定义了**两个内置对象，即 Global 和 Math** （它们也是本地对象，根据定义，每个内置对象都是本地对象）。

如此就可以理解了。内置对象是本地对象的一种。而其包含的两种对象中，Math对象我们经常用到，可这个Global对象是啥东西呢？

Global对象是ECMAScript中最特别的对象，因为实际上它根本不存在，有点玩人的意思。大家要清楚，在ECMAScript中，不存在独立的函数，所有函数都必须是某个对象的方法。

类似于isNaN()、parseInt()和parseFloat()方法等，看起来都是函数，而实际上，它们都是Global对象的方法。而且Global对象的方法还不止这些.

**宿主对象**: ECMAScript中的“宿主”就是我们网页的运行环境，即“操作系统”和“浏览器”。所有非本地对象都是宿主对象（host object），即由 ECMAScript 实现的宿主环境提供的对象。所有的BOM和DOM对象都是宿主对象。因为其对于不同的“宿主”环境所展示的内容不同。其实说白了就是，ECMAScript官方未定义的对象都属于宿主对象，因为其未定义的对象大多数是自己通过ECMAScript程序创建的对象。自定义的对象也是宿主对象。

1. **JQuery中有几种类型的选择器**

1.层叠选择器$(“form input”)

2.基本过滤选择器:first:last:not()

3.内容过滤选择器:odd:eq():animated

4.可视化过滤选择器:hidden:visible

5.属性过滤选择器：div[id]

6.子元素过滤选择器:first-child:last-child:only:child

7.表单元素过滤选择器:enabled:disabled:checked:selected

8.id,类，类型,元素...

1. **jQuery中的Delegate{}函数有什么作用**

delegate() 方法为指定的元素（属于被选元素的子元素）添加一个或多个事件处理程序，并规定当这些事件发生时运行的函数。

使用 delegate() 方法的事件处理程序适用于当前或未来的元素（比如由脚本创建的新元素）。$("div").delegate("button","click",function(){

$("p").slideToggle();

});

1. **用性能最高的一个方式写一个函数去掉数组重复元素，返回一**

**个新数组**

function unique(array){

var n = []; //一个新的临时数组

//遍历当前数组

for(var i = 0; i < array.length; i++){

//如果当前数组的第i已经保存进了临时数组，那么跳过，

//否则把当前项push到临时数组里面

if (n.indexOf(array[i]) == -1) {

n.push(array[i]);

}

}

return n;

}

// 判断浏览器是否支持indexOf ，indexOf 为ecmaScript5新方法 IE8以下（包括IE8， IE8只支持部分ecma5）不支持

if (!Array.prototype.indexOf){

// 新增indexOf方法

Array.prototype.indexOf = function(item){

var result = -1, a\_item = null;

if (this.length == 0){

return result;

}

for(var i = 0, len = this.length; i < len; i++){

a\_item = this[i];

if (a\_item === item){

result = i;

break;

}

}

return result;

}

}

1. **行内元素有那些。块级元素有那些。空元素有哪些**

首先：CSS规范规定，每个元素都有display属性，确定该元素的类型，每个元素都有默认的display值，如div的display默认值为“block”，则为“块级”元素；span默认display属性值为“inline”，是“行内”元素。

（1）行内元素有：a b span img input select strong（强调的语气）

（2）块级元素有：div ul ol li dl dt dd h1 h2 h3 h4…p

（3）常见的空元素：

<br> <hr> <img> <input> <link> <meta>

鲜为人知的是：

<area> <base> <col> <command> <embed> <keygen> <param> <source> <track> <wbr>

1. **说几条javasprit的基本规范**

1.不要在同一行声明多个变量。

2.请使用 ===/!==来比较true/false或者数值

3.使用对象字面量替代new Array这种形式

4.不要使用全局函数。

5.Switch语句必须带有default分支

6.函数不应该有时候有返回值，有时候没有返回值。

7.For循环必须使用大括号

8.If语句必须使用大括号

9.for-in循环中的变量 应该使用var关键字明确限定作用域，从而避免作用域污染。

1. **介绍一下标准的css盒子模型，低版本ie盒子的模型有什么不同**

（1）有两种， IE 盒子模型、W3C 盒子模型；

（2）盒模型： 内容(content)、填充(padding)、边界(margin)、 边框(border)；

（3）区 别： IE的width部分把 border 和 padding计算了进去;

1. **说出三种减少页面加载的方法（加载时间指感知的时间或实际**

**加载的时间）**

CSS Sprites；

JS、CSS源码压缩、图片大小控制合适；

网页Gzip；

CDN托管；

data缓存 ；

图片服务器；

1. **用js代码简单的介绍下自己**

<script>  
    function person(name,jingli,jineng) {  
      this.name=name;  
      this.jingli=jingli;  
      this.jineng=jineng;  
    }  
    person.prototype.show=function(){  
      console.log("我是"+this.name+"我有如下经历:"+this.jingli+"我会如下技能"+this.jineng);  
    }  
    var myself=new person("田野","小田工作室创办人，凤翔网络推广顾问","熟悉前端基本技能，熟悉网络营销思想有实战经验，掌握项目经理技能，可以编写文档，也可以使用axure进行原型设计，掌握自动化测试和性能测试技能")  
    myself.show();  
  </script>

1. **Html5中datalist是什么**

<datalist> 标签定义选项列表，与 input 元素配合使用该元素，来定义 input 可能的值。

datalist 及其选项不会被显示出来，它仅仅是合法的输入值列表。

<input id="myCar" list="cars" />

<datalist id="cars">

<option value="BMW">

<option value="Ford">

<option value="Volvo">

</datalist>

1. **Ajax同步和异步的区别，如何解决跨域问题**

同步的概念应该是来自于OS中关于同步的概念:不同进程为协同完成某项工作而在先后次序上调整(通过阻塞,唤醒等方式).同步强调的是顺序性.谁先谁后.异步则不存在这种顺序性.

同步：浏览器访问服务器请求，用户看得到页面刷新，重新发请求,等请求完，页面刷新，新内容出现，用户看到新内容,进行下一步操作。

异步：浏览器访问服务器请求，用户正常操作，浏览器后端进行请求。等请求完，页面不刷新，新内容也会出现，用户看到新内容。

jsonp、 iframe、window.name、window.postMessage、服务器上设置代理页面

1. **Html5中本地存储概念是什么，有什么优点**

cookie是网站为了标示用户身份而储存在用户本地终端（Client Side）上的数据（通常经过加密）。

cookie数据始终在同源的http请求中携带（即使不需要），记会在浏览器和服务器间来回传递。

sessionStorage和localStorage不会自动把数据发给服务器，仅在本地保存。

存储大小：

cookie数据大小不能超过4k。

sessionStorage和localStorage 虽然也有存储大小的限制，但比cookie大得多，可以达到5M或更大。

有期时间：

localStorage 存储持久数据，浏览器关闭后数据不丢失除非主动删除数据；

sessionStorage 数据在当前浏览器窗口关闭后自动删除。

cookie 设置的cookie过期时间之前一直有效，即使窗口或浏览器关闭

1. **什么是ajax和json，它们的优缺点**

ajax的全称：Asynchronous Javascript And XML。

异步传输+js+xml。

所谓异步，在这里简单地解释就是：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，页面是不会发生整页刷新的，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

1. **请使用javascript写出快速排序代码**

<script>

function quiktSort(arr){

var left =[],right=[];

if(arr.length<1){

return arr;

}

var index = Math.floor(arr.length/2);

var point = arr.splice(index,1);

for(var i=0,len=arr.length;i<len;i++){

if(arr[i]<point){

left.push(arr[i]);

}else{

right.push(arr[i]);

}

}

return quickSort(left).concat(point,quickSort(right));

}

</script>

1. **Html5有那些新增的表单元素**

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

1. **http状态码有那些，分别代表什么意思**

简单版:

100 Continue 继续，一般在发送post请求时，已发送了http header之后服务端将返回此信息，表示确认，之后发送具体参数信息

200 OK 正常返回信息

201 Created 请求成功并且服务器创建了新的资源

202 Accepted 服务器已接受请求，但尚未处理

301 Moved Permanently 请求的网页已永久移动到新位置。

302 Found 临时性重定向。

303 See Other 临时性重定向，且总是使用 GET 请求新的 URI。

304 Not Modified 自从上次请求后，请求的网页未修改过。

400 Bad Request 服务器无法理解请求的格式，客户端不应当尝试再次使用相同的内容发起请求。

401 Unauthorized 请求未授权。

403 Forbidden 禁止访问。

404 Not Found 找不到如何与 URI 相匹配的资源。

500 Internal Server Error 最常见的服务器端错误。

503 Service Unavailable 服务器端暂时无法处理请求（可能是过载或维护）。

完整版

1\*\*(信息类)：表示接收到请求并且继续处理

100——客户必须继续发出请求

101——客户要求服务器根据请求转换HTTP协议版本

2\*\*(响应成功)：表示动作被成功接收、理解和接受

200——表明该请求被成功地完成，所请求的资源发送回客户端

201——提示知道新文件的URL

202——接受和处理、但处理未完成

203——返回信息不确定或不完整

204——请求收到，但返回信息为空

205——服务器完成了请求，用户代理必须复位当前已经浏览过的文件

206——服务器已经完成了部分用户的GET请求

3\*\*(重定向类)：为了完成指定的动作，必须接受进一步处理

300——请求的资源可在多处得到

301——本网页被永久性转移到另一个URL

302——请求的网页被转移到一个新的地址，但客户访问仍继续通过原始URL地址，重定向，新的URL会在response中的Location中返回，浏览器将会使用新的URL发出新的Request。

303——建议客户访问其他URL或访问方式

304——自从上次请求后，请求的网页未修改过，服务器返回此响应时，不会返回网页内容，代表上次的文档已经被缓存了，还可以继续使用

305——请求的资源必须从服务器指定的地址得到

306——前一版本HTTP中使用的代码，现行版本中不再使用

307——申明请求的资源临时性删除

4\*\*(客户端错误类)：请求包含错误语法或不能正确执行

400——客户端请求有语法错误，不能被服务器所理解

401——请求未经授权，这个状态代码必须和WWW-Authenticate报头域一起使用

HTTP 401.1 - 未授权：登录失败

　　HTTP 401.2 - 未授权：服务器配置问题导致登录失败

　　HTTP 401.3 - ACL 禁止访问资源

　　HTTP 401.4 - 未授权：授权被筛选器拒绝

HTTP 401.5 - 未授权：ISAPI 或 CGI 授权失败

402——保留有效ChargeTo头响应

403——禁止访问，服务器收到请求，但是拒绝提供服务

HTTP 403.1 禁止访问：禁止可执行访问

　　HTTP 403.2 - 禁止访问：禁止读访问

　　HTTP 403.3 - 禁止访问：禁止写访问

　　HTTP 403.4 - 禁止访问：要求 SSL

　　HTTP 403.5 - 禁止访问：要求 SSL 128

　　HTTP 403.6 - 禁止访问：IP 地址被拒绝

　　HTTP 403.7 - 禁止访问：要求客户证书

　　HTTP 403.8 - 禁止访问：禁止站点访问

　　HTTP 403.9 - 禁止访问：连接的用户过多

　　HTTP 403.10 - 禁止访问：配置无效

　　HTTP 403.11 - 禁止访问：密码更改

　　HTTP 403.12 - 禁止访问：映射器拒绝访问

　　HTTP 403.13 - 禁止访问：客户证书已被吊销

　　HTTP 403.15 - 禁止访问：客户访问许可过多

　　HTTP 403.16 - 禁止访问：客户证书不可信或者无效

HTTP 403.17 - 禁止访问：客户证书已经到期或者尚未生效

404——一个404错误表明可连接服务器，但服务器无法取得所请求的网页，请求资源不存在。eg：输入了错误的URL

405——用户在Request-Line字段定义的方法不允许

406——根据用户发送的Accept拖，请求资源不可访问

407——类似401，用户必须首先在代理服务器上得到授权

408——客户端没有在用户指定的饿时间内完成请求

409——对当前资源状态，请求不能完成

410——服务器上不再有此资源且无进一步的参考地址

411——服务器拒绝用户定义的Content-Length属性请求

412——一个或多个请求头字段在当前请求中错误

413——请求的资源大于服务器允许的大小

414——请求的资源URL长于服务器允许的长度

415——请求资源不支持请求项目格式

416——请求中包含Range请求头字段，在当前请求资源范围内没有range指示值，请求也不包含If-Range请求头字段

417——服务器不满足请求Expect头字段指定的期望值，如果是代理服务器，可能是下一级服务器不能满足请求长。

5\*\*(服务端错误类)：服务器不能正确执行一个正确的请求

HTTP 500 - 服务器遇到错误，无法完成请求

　　HTTP 500.100 - 内部服务器错误 - ASP 错误

　　HTTP 500-11 服务器关闭

　　HTTP 500-12 应用程序重新启动

　　HTTP 500-13 - 服务器太忙

　　HTTP 500-14 - 应用程序无效

　　HTTP 500-15 - 不允许请求 global.asa

　　Error 501 - 未实现

HTTP 502 - 网关错误

HTTP 503：由于超载或停机维护，服务器目前无法使用，一段时间后可能恢复正常

1. **什么是闭包（closure）为什么要用它**

闭包是指有权访问另一个函数作用域中变量的函数，创建闭包的最常见的方式就是在一个函数内创建另一个函数，通过另一个函数访问这个函数的局部变量,利用闭包可以突破作用链域，将函数内部的变量和方法传递到外部。

闭包的特性：

1.函数内再嵌套函数

2.内部函数可以引用外层的参数和变量

3.参数和变量不会被垃圾回收机制回收

//li节点的onclick事件都能正确的弹出当前被点击的li索引

<ul id="testUL">

<li> index = 0</li>

<li> index = 1</li>

<li> index = 2</li>

<li> index = 3</li>

</ul>

<script type="text/javascript">

var nodes = document.getElementsByTagName("li");

for(i = 0;i<nodes.length;i+= 1){

nodes[i].onclick = (function(i){

return function() {

console.log(i);

} //不用闭包的话，值每次都是4

})(i);

}

</script>

执行say667()后,say667()闭包内部变量会存在,而闭包内部函数的内部变量不会存在

使得Javascript的垃圾回收机制GC不会收回say667()所占用的资源

因为say667()的内部函数的执行需要依赖say667()中的变量

这是对闭包作用的非常直白的描述

function say667() {

// Local variable that ends up within closure

var num = 666;

var sayAlert = function() {

alert(num);

}

num++;

return sayAlert;

}

var sayAlert = say667();

sayAlert()//执行结果应该弹出的667

1. **你知道那些针对jQuery的优化方法**

基于Class的选择性的性能相对于Id选择器开销很大，因为需遍历所有DOM元素。

频繁操作的DOM，先缓存起来再操作。用Jquery的链式调用更好。

比如：var str=$("a").attr("href");

for (var i = size; i < arr.length; i++) {}

for 循环每一次循环都查找了数组 (arr) 的.length 属性，在开始循环的时候设置一个变量来存储这个数字，可以让循环跑得更快：

for (var i = size, length = arr.length; i < length; i++) {}

1. **用原型链继承的方式写一个类和子类**

function Person(name,age){

this.name=name;

this.age=age;

}

Person.prototype.study=function(){

return "学习"

}

/\*var p1 =new Person("张三",20);\*/

/\*p1.study();\*/

function Student(class\_,name,age){

this.class\_=class\_;

this.name=name;

this.age=age;

}

Student.prototype=new Person();

var s1 =new Student("二班","李大人",16);

console.log(s1.name,s1.age,s1.class\_,s1.study());

1. **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 num(str) {

var num1 = str.length;

var num2 = 0;

for (var i = 0; i < str.length; i++) {

if (str.charCodeAt(i) >= 10000) {

num2++;

}

}

console.log(num1 + num2)

}

1. **简单概括浏览器事件模型，如何获得资源dom节点**

浏览器事件模型分为三个阶段

1、捕获阶段

2、目标阶段

3、冒泡阶段

1. **写一段ajax提交的js代码**

var xhr =xhr();

function xhr(){

if(window.XMLHttpRequest){

return window. XMLHttpRequest();

}else if(window.ActiveXObject){

try {

return new ActiveXObject("Microsoft.XMLHTTP");

}catch (e) {

try {

return new ActiveXObject("Msxml2.XMLHTTP");

}catch (ex) { }

}

}

}

xhr.open("get","url","true");

xhr.onreadystatechange=function()

{

if (xhr.readyState==4 && (xhr.status==200||xhr.status==304))

{

document.getElementById("myDiv").innerHTML=xhr.responseText;

}

}

xhr.send();

1. **判断字符串是否是这样组成的，第一个必须是字母，后面可以**

**是字母和数字、下划线总长度为5-20（请使用正则表达式）**

function if\_fit(str){

var reg=/^[A-Za-z]{1}\w{5,20}/g;

var result=str.search(reg);

return result;

}

1. **截取字符串abcdefg的efg**

var str="abcdefg";

console.log(str.slice(4));

1. **在css引入的方式有那些，link和@import的区别是什么**

内联方式、嵌入方式、链接方式、导入方式

　　区别1：link是XHTML标签，除了加载CSS外，还可以定义RSS等其他事务；@import属于CSS范畴，只能加载CSS。

　　区别2：link引用CSS时，在页面载入时同时加载；@import需要页面网页完全载入以后加载。

　　区别3：link是XHTML标签，无兼容问题；@import是在CSS2.1提出的，低版本的浏览器不支持。

　　区别4：link支持使用Javascript控制DOM去改变样式；而@import不支持。

1. **将字符串helloChina反转输出**

var str = "helloChina";

方法1：console.log( str.split("").reverse().join("") )；');

方法2：for (var x = str.length-1; x >=0; x--)

{

document.write(str.charAt(x));

}

方法3：var a=str.split("");

var rs = new Array;

while(a.length)

{

rs.push(a.pop());

}

alert(rs.join(""));

1. **为什么无法定义1px左右高度的容器**

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

1. **FireFox中标签的居中问题的解决办法**

\*{margin：0px auto；}

1. **请写出XHTML和css如何注释**

XHTML:<!-- 注释内容-->

css:/\* 注释内容\*/

1. **现在想调节一下父元素的透明度，但是又不影响子元素的透明**

**度，怎么破**

方法1：用RGBA

方法2：再加一层与父元素同级的div装载子元素 定位到子元素原位置

1. **简述cookies sessionStorage 和localStorage的区别**

区别：cookie数据始终在同源的http请求中携带（即使不需要），即cookie在浏览器和服务器间来回传递。而sessionStorage和localStorage不会自动把数据发给服务器，仅在本地保存。cookie数据还有路径（path）的概念，可以限制cookie只属于某个路径下。存储大小限制也不同，cookie数据不能超过4k，同时因为每次http请求都会携带cookie，所以cookie只适合保存很小的数据，如会话标识。sessionStorage和localStorage 虽然也有存储大小的限制，但比cookie大得多，可以达到5M或更大。数据有效期不同，sessionStorage：仅在当前浏览器窗口关闭前有效，自然也就不可能持久保持；localStorage：始终有效，窗口或浏览器关闭也一直保存，因此用作持久数据；cookie只在设置的cookie过期时间之前一直有效，即使窗口或浏览器关闭。作用域不同，sessionStorage不在不同的浏览器窗口中共享，即使是同一个页面；localStorage 在所有同源窗口中都是共享的；cookie也是在所有同源窗口中都是共享的。Web Storage 支持事件通知机制，可以将数据更新的通知发送给监听者。Web Storage 的 api 接口使用更方便。

1. **简述ECMASCRIPT6的新特性**

1.增加块作用域

2.增加let const

3.解构赋值

4.函数参数扩展 （函数参数可以使用默认值、不定参数以及拓展参数）

5.增加class类的支持

6.增加箭头函数

7.增加模块和模块加载（ES6中开始支持原生模块化啦）

8.math, number, string, array, object增加新的API

1. **Apply和call方法的异同**

相同点:两个方法产生的作用是完全一样的，第一个参数都是对象；

不同点:

call()方法参数将依次传递给借用的方法作参数，即fn.call(thisobj, arg1,arg2,arg3...argn)，有n个参数

apply()方法第一个参数是对象，第二个参数是数组fn.apply(thisobj,arg)，此处的arg是一个数组,只有两个参数

1. **在javascript中什么是伪数组，如何将伪数组转化为标准数组**

这里把符合以下条件的对象称为伪数组：

1，具有length属性

2，按索引方式存储数据

3，不具有数组的push,pop等方法

伪数组（类数组）：无法直接调用数组方法或期望length属性有什么特殊的行为，不具有数组的push,pop等方法，但仍可以对真正数组遍历方法来遍历它们。典型的是函数的argument参数，还有像调用getElementsByTagName,document.childNodes之类的,它们都返回NodeList对象都属于伪数组。可以使用Array.prototype.slice.call(fakeArray)将数组转化为真正的Array对象。

1. **Js和nitive交互的方法与问题**

**实现JS和Native有两种方式:**

js与java互相调用

**第一种：shouldOverrideUrlLoading(WebView view, String url）**

通过给WebView加一个事件监听对象（WebViewClient)并重写shouldOverrideUrlLoading(WebView view, String url）方法。当按下某个连接时WebViewClient会调用这个方法，并传递参数view和url

**第二种：JS和Java互调**

WebView开启JavaScript脚本执行

WebView设置供JavaScript调用的交互接口

客户端和网页端编写调用对方的代码

JS调用JAVA

JS : window.jsInterfaceName.methodName(parameterValues)

native: webView.addJavascriptInterface(new JsInteration(), “androidNative”);

下面给出一个实例,方便理解

webView.addJavascriptInterface(new JsInteration(), “androidNative”);

@JavascriptInterfacepublic void helloJS(){…}

window.androidNative.helloJS();

Java调用JS

webView调用js的基本格式为webView.loadUrl(“javascript:methodName(parameterValues)”)

调用js无参无返回值函数： String call =“javascript:sayHello()”；webView.loadUrl(call);

调用js有参无返回值函数：String call = “javascript:alertMessage(\”” + “content” + “\”)”; webView.loadUrl(call);

调用js有参数有返回值的函数

Android在4.4之前并没有提供直接调用js函数并获取值的方法，所以在此之前，常用的思路是 java调用js方法，js方法执行完毕，再次调用java代码将值返回。

Android 4.4之后使用evaluateJavascript即可。

private void testEvaluateJavascript(WebView webView) {

webView.evaluateJavascript("getGreetings()", new ValueCallback<String>() {

@Override

public void onReceiveValue(String value) {

Log.i(LOGTAG, "onReceiveValue value=" + value);

}});

}

注:

参数类型如果是简单的int或String，可以直接传，对于复杂的数据类型，建议以字符串形式的json返回。

evaluateJavascript方法必须在UI线程（主线程）调用，因此onReceiveValue也执行在主线程。

当native与js交互时存cookie看到很多人遇到过这样一个问题,cookie存不进去,网上有很多解释方案,但是很多没说到重点上,这里直接贴一下代码:

public static void synCookies(Context context, String url, String version) {

CookieSyncManager.createInstance(context);

CookieManager cookieManager = CookieManager.getInstance();

cookieManager.setAcceptCookie(true);

cookieManager.removeAllCookie();

cookieManager.setCookie(url, "sessionKey=" + UserInfoShareprefrence.getInstance(context).getLocalSessionKey());

cookieManager.setCookie(url, "productVersion=android-epocket-v" + version);

CookieSyncManager.getInstance().sync();

}

存不进去的很大一部分原因是你的url不对,他官方给出的解释是这样的

/\*\*

\* Sets a cookie for the given URL. Any existing cookie with the same host,

\* path and name will be replaced with the new cookie. The cookie being set

\* will be ignored if it is expired.

\*

\* @param url the URL for which the cookie is to be set

\* @param value the cookie as a string, using the format of the 'Set-Cookie'

\* HTTP response header

\*/

public void setCookie(String url, String value) {

throw new MustOverrideException();

}

其实没说明白url到底是什么,这里的url就是显示的url的域名,这里顺便贴出取域名的方法,给出的是通过正则提取域名

/\*\*

\* 获得域名

\*

\* @param url

\* @return

\*/public static String getDomain(String url) {

Pattern p = Pattern.compile("[^//]\*?\\.(com|cn|net|org|biz|info|cc|tv)", Pattern.CASE\_INSENSITIVE);

Matcher matcher = p.matcher(url);

matcher.find();

return matcher.group();

}

还有一点就是,如果你想传递多个值给cookie的话,可以多次使用setCookie,不要擅自的自己拼值,因为你拼的字符串中可能存在分号,内部多分号做了特殊处理,截取分号之前的,之后的直接放弃!

1. **用sass的minix定义一些代码片段，且可传参数**

/\*\*

\* @module 功能

\* @description 生成全屏方法

\* @method fullscreen

\* @version 1.7.0

\* @param {Integer} $z-index 指定层叠级别 <1.7.0>

\* @param {Keywords} $position 指定定位方式，取除`static | relative`之外的值，默认值：absolute <1.8.5>

\*/

@mixin fullscreen($z-index: null, $position: absolute) {

   position: $position;

   z-index: $z-index;

   top: 0;

   right: 0;

   bottom: 0;

   left: 0;

}

1. **移动端经常出现的兼容问题，谈谈移动端应用或者wap站得**

**一些优化技巧和心得**

http://www.jb51.net/article/84973.htm

1. **H5中新增的一些单位rem是什么意思，和em的关系，以及rem在自适应布局中的应用方法**

答：Em为单位：

这种技术需要一个参考点，一般都是以<body>的“font-size”为基准。比如说我们使用“1em”等于“10px”来改变默认值“1em=16px”，这样一来，我们设置字体大小相当于“14px”时，只需要将其值设置为“1.4em”。

Rem为单位：

rem是相对于根元素<html>的“font-size”为基准。比如说我们给html设置font-size为100px，

那么我们要给html中的p标签设置16px的字体，font-size设置.16rem就可以，在这里16px=.16rem。

这个单位与em有什么区别呢？

区别在于使用rem为元素设定字体大小时，仍然是相对大小，但相对的只是HTML根元素。这个单位可谓集相对大小和绝对大小的优点于一身，通过它既可以做到只修改根元素就成比例地调整所有字体大小，又可以避免字体大小逐层复合的连锁反应。目前，除了IE8及更早版本外，所有浏览器均已支持rem。对于不支持它的浏览器，应对方法也很简单，就是多写一个绝对单位的声明。这些浏览器会忽略用rem设定的字体大小。

1. **如何实现浏览器内多个标签页之间的通信？**

通过WebSocket或SharedWorker把客户端和服务器端建立socket连接，从而实现通信；也可以调用localstorge、cookies等本地存储方法。

1. **假设现在页面里有一个id是con的div，现在需要编写js代码，在页面加载完成后 将div的高度设置成100px，宽度设置成60px，并设置成灰色的ipx的边框，北京设置成浅黄色。**

window.onload=function(){

var oDiv=document.getElementById("con");

oDiv.style.height="100px";

oDiv.style.width="60px";

oDiv.style.width="1px solid gray";

oDiv.style.color="yellow";

}

1. **对新技术有那些了解，常去的网站有那些**

掘金、简书、github、csdn等

1. **列举几种后端通讯的方法，分别使用的场景**

答：1.后端程序可以通过session来进行通讯，session有过期时间，主要用于验证码的验证，登录过期等的应用。

2.数据库，数据库支持多种语言的操作，那么通过数据库就可以通讯。

1. **用程序找出数组中出现次数超过一半的数字**

思路：

    1、 一个数字在数组中出现次数超过了一半，则排序后，位于数组中间的数字一定就是该出现次数超过了长度一半的数字（可以用反证法证明），也即是说，这个数字就是统计学上的中位数。最容易想到的办法是用快速排序对数组排序号后，直接取出中间的那个数字，这样的时间复杂度为O(nlogn)，空间复杂度为O(1)。

    2 、事实上可以不用对数组进行排序，或者说仅部分排序，受快速排序的partition函数的启发，我们可以利用反复调用partition函数来求的该数字。我们现在数组中随机选取一个数字，而后通过Partition函数返回该数字在数组中的索引index，如果index刚好等于n/2，则这个数字便是数组的中位数，也即是要求的数，如果index大于n/2，则中位数肯定在index的左边，在左边继续寻找即可，反之在右边寻找。这样可以只在index的一边寻找，而不用两边都排序，减少了一半排序时间。这种情况的平均时间复杂度大致为：T(n) = n+n/2+n/4+n/8+....+1，很明显当n很大时，T(n)趋近于2n，也就是说平均情况下时间复杂度为O(n),但是这种情况下，最坏的时间复杂度依然为O(n\*n)，最坏情况下，index总是位于数组的最左或最右边，这样时间复杂度为T(n) = n+n-1+n-2+n-3+....+1 = n(n-1)/2，显然，时间复杂度为O(n\*n)，空间复杂度为O(1)。

1. **请设计一套方案，用于确保页面中js加载完全，对于优化某网页的加载速度，有什么独到见解**

答：js方法：

<script type="text/javascript">

window.onload=function(){

var userName="xiaoming";

alert(userName);

}

</script>

jquery方法：

<script type="text/javascript">

$(document).ready(function(){

var userName="xiaoming";

alert(userName);

});

</script>

或者简写：

$(function(){

var userName="xiaoming";

alert(userName);

});

如何确定一个js是否加载完全或者页面中的所有js加载完全，具体办法如下：

function loadScript（ url, callback） {

var script = document.createElement("script");

script.type = "text/javascript";

if (script.readyState) {

script.onreadystatechange = function() {

if (script.readyState == "loaded" || script.readyState == "complete") {

script.onreadystatechange = null;

callback();

}

}

} else {

script.onload = function() {

callback();

}

}

script.src = url;

document.getElementsByName("head")[0].appendChild(script);

}

如何让脚本的执行顺序按照你设定的顺序执行，使用嵌套的方式：

loadScript("file1.js", function() {

loadScript("file2.js", function() {

loadScript("file3.js", function() {

alert("All files are loaded");

});

});

});

**网页加载速度优化：**

**1、减少请求**

最大的性能漏洞就是一个页面需要发起几十个网络请求来获取诸如样式表、脚本或者图片这样的资源，这个在相对低带宽和高延迟的移动设备连接上来说影响更严重。

CDNs（内容分发网络）把资源放在离用户地理位置更近的地方对解决这个问题能起到很大作用，但是比起获取请求，大量的请求对页面加载时间的影响更为严重，而且最近的发现表明，CDNs对移动端用户的性能影响越来越低。

**2、整合资源**

对开发者来说，将Javascript代码和CSS样式放到公共的文件中供多个页面共享是一种标准的优化方法，这个方法能很简单的维护代码，并且提高客户端缓存的使用效率。

在Javascript文件中，要确保在一个页面中相同的脚本不会被加载多次，当大团队或者多个团队合作开发的时候，这种冗余的脚本就很容易出现，你可能会对它的发生频率并不低感到非常吃惊。

Sprites是css中处理图片的一项技术，Sprites就是将多张图片整合到一个线性的网状的大图片中，页面就可以将这个大图片一次性获取回来并且做为css的背景图，然后使用css的背景定位属性展示页面需要的图片部分，这种技术将多个请求整合成一个，能显著地改善性能。

平稳地改进但是需要对资源有控制权限，根据开发者的网站不同权限，一些资源并不需要被整合起来（例如，一些由CMS生成的资源），还有，对于一些外部域引用的资源，强行整合可能会导致问题，马海祥提醒大家需要注意的是，整合资源对手机浏览器来说是一把双刃剑，整合资源确实会在首次访问减少请求，但是大的资源文件可能会导致缓存失效，所以，需要小心地使用各种技术整合资源，以达到优化本地存储的目的。

**3、使用浏览器缓存和本地缓存**

现在所有的浏览器都会使用本地资源去缓存住那些被Cache-Control或者Expires头标记的资源，这些头能标记资源需要缓存的时间，另外，ETag（实体标签）和Last-Modified头来标识当资源过期后是否需要重新请求，浏览器为了减少不必要的服务器请求，尽可能地从本地缓存中获取资源，并且将那些已经过期的、或者当缓存空间减小的时候将那些很久不用的资源进行清理，浏览器缓存通常包括图片，CSS，Javascript代码，这些缓存能合理地提高网站的性能（比如为了支持后退和前进的按钮，使用一个单独的缓存来保存整个渲染的页面）。

移动浏览器缓存，通常是比桌面PC小的多，这就导致了缓存的数据会很经常被清理，HTML5的缓存基于浏览器缓存提供了一个很好的替换方案，Javascript的localStorage已经在所有主流的桌面和移动端浏览器上都实现了，使用脚本代码能简便地支持HTML5的localStorage操作，可以读写键值数据，每个域名大概有5MB的容量，虽然不同的移动浏览器上读写速度相差很大，但是localStorage大容量的缓存使得它很适合作为客户端的缓存，从localStorage获取资源明显快于从服务器上获取资源，而且在大多数移动设备上也比依靠缓存头或者浏览器的本地缓存更灵活可靠，这是移动浏览器比桌面PC更有优势的一个地方，在桌面PC上，本地缓存仍然优先使用标准的浏览器缓存，导致桌面PC本地缓存的性能落后于移动浏览器。

在此，马海祥要提醒各位一下：虽然localStorage的机制易于实现，但是它的一些控制机制却是非常复杂的，你需要考虑到缓存带给你的所有问题，比如缓存失效（什么时候需要删除缓存？），缓存丢失（当你希望数据在缓存中的时候它并不在怎么办？），还有当缓存满的时候你怎么办？

**4、首次使用的时候在HTML中嵌入资源**

HTML的标准是使用链接来加载外部资源，这使得更容易在服务器上（或者在CDN上）操作更新这些资源，而不是在每个页面上修改更新这些资源，根据上文讨论的，这种模式也使得浏览器能从本地缓存而不是服务器上获取资源。

但是对还没有缓存到浏览器localStorage的资源来说，这种模式对网站的性能有负面的影响，一般来说，一个页面需要几十个单独的请求来获取资源从而渲染页面。

所以说，从性能的角度来说，如果一个资源没有很高的被缓存的几率的话，最好把它嵌入到页面的HTML中（叫inlining），而不是使用链接外部，脚本和样式是支持内嵌到HTML中的，但是图片和其他的二进制资源其实也是可以通过内嵌包含base64编码的文本来嵌入到HTML中的。

内嵌的缺点是页面的大小会变得非常大，所以对于Web应用来说，关键的是能够跟踪分析这个资源什么时候需要从服务端获取，什么时候已经缓存到客户端了。

另外，在第一次请求资源后必须能够使用代码在客户端缓存资源，因此，在移动设备上，使用HTML5 localStorage能很好地做到内嵌。

由于不知道用户是否已经访问过这个页面了，所以需要网站有机制能生成不同版本的页面。

**5、使用HTML5服务端发送事件**

Web应用已经使用了各种从服务器上轮询资源的方法来持续地更新页面，HTML5的EventSource对象和Server-Sent事件能通过浏览器端的JavaScript代码打开一个服务端连接客户端的单向通道，服务端可以使用这个写通道来发送数据，这样能节省了HTTP创建多个轮询请求的消耗。

这种方式比HTML的WebSocket更高效，WebSocket的使用场景是，当有许多客户端和服务端的交互的时候（比如消息或者游戏），在全双工连接上建立一个双向通道。

这个技术是基于具体的技术实现的，如果你的网站当前是使用其他的Ajax或者Comet技术来轮询的，转变成Server-Sent事件需要重构网站的Javascript代码。

**6、消除重定向**

当用户在一个移动设备上访问桌面PC网站的时候，Web网站应用通常读取HTTP的user-agent头来判断这个用户是否是来自移动设备的，然后应用会发送带有空HTTP body和重定向HTTP地址头的HTTP 301（或者302）请求，把用户重定向到网站的移动版本上去，但是这个额外的客户端和服务端的交互通常在移动网络上会消耗几百毫秒，因此，在原先的请求上传递移动的web页会比传递一个重定向的信息并让客户端再请求移动页面更快。

对于那些想要在移动设备上看桌面PC网站的用户来说，你可以在移动web页面上提供一个链接入口，这样也能同时表示你的网站是并不提倡这种行为的。

虽然这个技术在理论上是简单的，但是实际上并不易于实施，由于有些m.sites是宿主在其他地方的，所以许多网站会选择重定向到一个不同的服务器上，有的网站则是会在重定向请求的时候种植上Cookie告诉Web应用这个用户是在使用移动设备，这种方法可能对web应用来说更容易控制。

**7、减少资源负载**

关于移动端页面的大小问题，渲染小页面更快，获取小资源也更快，减小每个请求的大小通常不如减少页面请求个数那么显著地提高性能。

但是有些技术在性能方面，特别是在需要对带宽和处理器性能精打细算的移动设备环境下，仍然是能带来很大利益的。

**8、压缩文本和图像**

诸如gzip这样的压缩技术，依靠增加服务端压缩和浏览器解压的步骤，来减少资源的负载，但是，一般来说，这些操作都是被高度优化过了，而且测试表明，压缩对网站还是起到优化性能的作用的，那些基于文本的响应，包括HTML，XML，JSON（Javascript Object Notation），Javascript，和CSS可以减少大约70%的大小。

浏览器在Accept-Encoding请求头中申明它的解压缩技术，并且当它们接收到服务端返回的Content-Encoding响应头标示的时候，就会按照这个响应头自动做解压操作。

马海祥觉得这种方法的优点就是易于实现，如果设置正确的话，现在所有的Web服务器都支持压缩响应，但是，也有一些桌面PC的安全工具会将请求头中的Accept-Encoding头去掉，这样即使浏览器支持解压缩，用户也无法获取到压缩后的响应。

**9、代码简化**

简化通常是使用在脚本和样式文件中，删除一些不必要的字符，比如空格，换行符，或者注释等，不需要暴露给外部的命名就可以被缩短为一个或者两个字符，比如变量名，合适的简化资源通常在客户端不需要做任何其他的处理，并且平均减少20%的资源大小，内嵌在HTML中的脚本和样式文件也是可以精简的，有很多很好的库来做精简化的操作，这些库一般也同时会提供合并多个文件这样减少请求数的服务（具体可查看马海祥博客《手机网站制作的常用方法及优化技巧》的相关介绍）。

简化带来的好处并不局限于减少带宽和延迟，对于那些移动设备上缓存无法保存的过大资源来说，也是很有改善的，Gzip在这个方面并没有任何帮助，因为资源是在被解压后才被缓存起来的。

Google的Closure Compiler已经难以置信地完成了理解和简化Javascript的工作，但是CSS的简化则没有那么容易，因为对不同浏览器来说有不同的CSS技术能迷惑CSS简化工具，然后让CSS简化后无法正常工作，马海祥提醒大家必须要注意的是，已经有这样的案例了，即使只是删除了不必要的字符，简化工作也有可能破坏页面，所以当你应用简化技术之后，请做一下完整的功能测试工作。

**10、调整图片大小**

图片通常是占用了Web页面加载的大部分网络资源，也占用了页面缓存的主要空间，小屏幕的移动设备提供了通过调整图片大小来加速传输和渲染图片资源的机会，如果用户只是在小的移动浏览器窗口中看图片的话，高分辨率的图片就会浪费带宽、处理时间和缓存空间。

为了加速页面渲染速度和减少带宽及内存消耗，可以动态地调整图片大小或者将图片替换为移动设备专用的更小的版本，不要依靠浏览器来将高分辨率的图片转换成小尺寸的图片，这样会浪费带宽。

另外一个方法是先尽快加载一个低分辨率的图片来渲染页面，在onload或者用户已经开始和页面交互以后将这些低分辨率的图片替换成为高分辨率的图片。

特别应用在高度动态化的网站是有优势的。

**11、使用HTML5和CSS 3.0来简化页面**

HTML5包括了一些新的结构元素，例如header，nav，article和footer，使用这些语义化的元素比传统的使用div和span标签能使得页面更简单和更容易解析，一个简单的页面更小加载更快，并且简单的DOM（Document Object Model）代表着更快的JavaScript执行效率，新的标签能很快地应用在包括移动端的新浏览器版本上，并且HTML5设计让那些不支持它的浏览器能平稳过渡使用新标签。

HTML5的一些表单元素提供了许多新属性来完成原本需要javascript来完成的功能，例如，新的placeholder属性用于显示在用户输入进入输入框之前显示的介绍性文字，autofocus属性用于标示哪个输入框应当被自动定位。

也有一些新的输入框元素能不用依靠Javascript就可以完成一些通用的需求，这些新的输入框类型包括像e-mail，URL，数字，范围，日期和时间这样需要复杂的用户交互和输入验证的元素，在移动浏览器上，当需要输入文本的时候，弹出的键盘通常是由特定的输入框类型来做选择的，不支持指定的输入类型的浏览器就会只显示一个文本框。

另外，只要浏览器支持内建的层次，圆角，阴影，动画，过渡和其他的图片效果，CSS 3.0就能帮助你创建轻便简易的页面了，而这些图片效果原先是需要加载图片才能完成的，这样，这些新特性就能加速页面渲染了。

人工地做这些改动是非常复杂和耗时的，如果你使用CMS，它可以帮你生成许多你不需要控制的HTML和CSS（具体可查看马海祥博客《制作移动端手机网站过程中的SEO优化方法技巧》的相关介绍）。

**12、延迟渲染”BELOW-THE-FOLD”内容**

可以确定的是如果我们将不可见区域的内容延迟加载，那么页面就会更快地展现在用户面前，这个区域叫做“below the fold”，为了减少页面加载后需要重新访问的内容，可以将图片替换为正确的高宽所标记的<img>标签。

一些好的Javascript库可以用来处理这些below-the-fold 延迟加载的图像。

**13、延迟读取和执行的脚本**

在一些移动设备上，解析Javascript代码的速度能达到100毫秒每千字节，许多脚本的库直到页面被渲染以后都是不需要的加载的，下载和解析这些脚本可以很安全地被推迟到onload事件之后来做。

例如，一些需要用户交互的行为，比如托和拽，都不大可能在用户看到页面之前被调用，相同的逻辑也可以应用在脚本执行上面，尽量将脚本的执行延迟到onload事件之后，而不是在初始化页面中重要的可被用户看到的内容的时候执行。

这些延迟的脚本可能是你自己写的，更重要的是，也有可能是第三方的，对广告、社交媒体部件、或者分析的差劲的脚本优化会导致阻塞页面的渲染，会增加珍贵的加载时间，当然，你需要小心地评估诸如jquery这样为移动网站设计的大型脚本框架，特别当你仅仅只是使用这些框架中的一些对象的时候更要小心评估。

许多第三方的框架现在提供延迟加载的异步版本的API，开发者只需要将原先的逻辑转化到这个异步版本，一些JavaScript要做延迟加载会有些复杂，因为在onload之后执行这些脚本需要注意很多注意事项（例如，你有个脚本需要绑定到onload事件上，你需要做什么？如果你将脚本延迟到onload事件之后，就一定就会失去很多执行的时机）。

**14、使用Ajax来增强进程**

Ajax（Asynchronous JavaScript and XML）是一项使用XHR（XMLHttpRequest）对象来从Web服务器上获取数据的技术，它并不需要更新正在运行的页面，Ajax能更新页面上的某个部分而不需要重新构建整个页面，它通常用来提交用户的交互相应，但是也可以用来先加载页面的框架部分，然后当用户准备好浏览网页的时候再填充详细的内容。

尽管是这个名字，但是XMLHttpRequest并不强制要求你只能使用XML，你可以通过调用overrideMineType方法来制定“application/json”类型来使用json替换XML，使用JSON.parse会比使用原生的eval()函数快了几乎两倍，并且更为安全。

同时，切记Ajax的返回响应也会得益于那些应用在普通的返回响应的优化技术上面，确保对你的Ajax返回响应使用了缓存头，简化，gzip压缩，资源合并等技术。

由于这个技术是根据具体应用不同而不同的，所以很难量化，或许由于跨域问题，你需要使用XHR2，这个技术能使用外部域的资源，从而能进行跨域的XHR请求。

**15、根据网络状况进行适配处理**

由于使用更多带宽会使用更多移动网络的费用，所以只有能检测网络的类型才能使用针对特定网络的优化技术。

例如，预加载未来使用到的请求是非常聪明的做法，但是如果用户的带宽很稀有，并且加载的有些资源是永远不会用到的话，这个技术就是不合理的了。

在Android 2.2+，navigator.connection.type属性的返回值能让你区分Wifi和2G/3G/4G网络，在Blackberry上，blackberry.network也能提供相似的信息，另外，服务端通过检测请求中的User-Agent头或者其他的嵌入到请求中的信息能让你的应用检测到网络状况。

检测网络信息的API最近已经有所变化了，接口现在不是直接定义Wi-Fi，3G等网络状况，而是给出了带宽信息和诸如“非常慢，慢，快和非常快”这样的建议，有个属性能给出估计的MB/s值和一个“meterd”的Boolean值来表示它的可信度，但是对浏览器来说，很难根据这个来判断环境，判断当前网络环境然后适配仍然是一种最好的方法（具体可查看马海祥博客《百度移动搜索开放适配服务的3种方法》的相关介绍），但是这种方法正在被考虑被替换。

**16、对多线程来说尽量使用HTML5的WEB WORKER特性**

HTML5中的Web Worker是使用多个线程并发执行Javascript程序，另外，这种特别的多线程实现能减少困惑开发者多年的，在其他平台上遇到的问题，例如，当一个线程需要改变一个正在被其他线程使用的资源该如何处理，在Web Worker中，子线程不能修改主用户界面（UI）线程使用的资源。

对提高移动站点的性能来说，Web Worker中的代码很适合用来预处理用户完成进一步操作所需要的资源的，特别是在用户的带宽资源不紧缺的情况下，在低处理器性能的移动设备上，过多的预加载可能会干扰当前页面的UI响应，使用多线程代码，让Web Worker对象（并且尽可能使用localStorage来缓存数据）在另外一个线程中操作预加载资源，这样就能不影响当前的UI表现了。

要特别说明的是，Web Worker只在Android 2.0以上的版本实现，而且iphone上的ios5之前的版本也不支持，在桌面PC上，总是落后的IE只在IE 10才支持Web Worker。

虽然这项技术并不是非常难实现，但是对Web Workers来说，有一些限制需要强制遵守，Web Workers不能进入到页面的DOM，也不能改变页面上的任何东西，Web Worker很适合那种需要后台计算和处理的工作。

**17、将CLICK事件替换成TOUCH事件**

在触摸屏设备上，当一个用户触碰屏幕的时候，onclick事件并没有立即触发，设备会使用大约半秒（大多数设备差不多都是300毫秒）来让用户确定是手势操作还是点击操作，这个延迟会很明显地影响用户期望的响应性能，要使用touchend事件来替换才能解决，当用户触碰屏幕的时候，这个事件会立即触发。

为了要确保不会产生用户不期望的行为，你应该也要使用touchstart和touchmove事件，例如，除非同时有个touchstart事件在button上，否则不要判断touchend事件在button上就意味着点击行为，因为用户有可能从其他地方触碰开始，然后拖拽到button上触碰结束的，你也可以在touchstart事件之后使用touchmove事件来避免将touchend事件误判为点击，当然前提是需要假设拖拽的手势并不是预期产生点击行为。

另外，你也需要去处理onclick事件来让浏览器改变button的外观从而标识为已点击的状态，同时你也需要处理那些不支持touch事件的浏览器，为了避免代码在touchend和onclick代码中重复执行，你需要在确保用户触碰事件已经在touchend执行了之后，在click事件中调用preventDefault和stopPropagation方法。

这种技术需要更多工作才能在一个页面中增加和维护链接，touch事件的代码必须考虑其他手势，因为替换click的还有可能是缩放或者敲击动作。

**18、支持SPDY协议**

应用层HTTP和HTTPS协议导致的一些性能瓶颈，使得不论是桌面还是移动端的网站都非常难受，在2009年，谷歌开始研发一种叫做SPDY（谐意是“speedy”）的协议来替换已有的协议，这种协议宣称能突破这些限制，这个协议的目标是让多种浏览器和多种Web服务都能支持，所以这个协议是开源的，但是初步地，只有Google的Chrome浏览器（在版本10及之后的）和google的站点支持，一旦一个Web服务支持SPDY，那么它上面的所有站点都可以和支持这个协议的浏览器使用SPDY进行交互，将SPDY应用在25个top100的Internet网站上，Google收集到的数据是网站的速度会改善27%到60%不等。

SPDY自动使用gzip压缩所有内容，和HTTP不同的是，它连header的数据也使用gzip压缩，SPDY使用多线程技术让多个请求流或者响应流能共用一个TCP连接，另外SPDY允许请求设置优先级，比如，页面中心的视频会比边框的广告拥有更高的优先级。

或许SPDY中最变革性的发明就是流是双向的，并且可以由客户端或者服务端发起，这样能使得信息能推送到客户端，而不用由客户端发起第一次请求，例如，当一个用户第一次浏览一个站点，还没有任何站点的缓存，这个时候服务端就可以在响应中推送所有的请求资源，而不用等候每个资源被再次独立请求了，作为替换协议，服务端可以发送暗示给客户端，提示页面需要哪些资源，同时也允许由客户端来初始化请求。即使是使用后一种这样的方式也比让客户端解析页面然后自己发现有哪些资源需要被请求来得快。

虽然SPDY并没有对移动端有什么特别的设置，但是移动端有限的带宽就使得如果支持SPDY的话，SPDY在减少移动网站的延迟是非常有用的。

依据网站和服务的环境来进行平稳操作或进一步考虑，Google有一个SPDY模块支持Apache2.2 – mod\_spdy – 这个模块是免费的；但是mod\_spy有线程上的问题，并且和mod\_php协作并不是很好，所以要求你使用这个技术的时候要确保你的网站的正常运行。

1. **请事先鼠标点击中的任意标签，alert该标签的名称（注意兼容性）**

function elementName(evt){

evt = evt|| window.event;

var selected = evt.target || evt.srcElement;

alert(selected.tagName);

}

1. **对string对象进行扩展，使其具有删除前后空格的方法**

String.prototype.trim = function() {

return this.replace(/(^\s\*)|(\s\*$)/g, "");

}

1. **常使用的库有哪些？常用的前端开发工具？开发过什么应用**

**或组件？**

1)bootstrap, easy UI, jqueryUI , jquery、angular.js, vue.js等。

2)前端开发工具：gulp webpack

1. **用一句话概述您的有点，用一句话概述您的缺点**

自由发挥

1. **描述下你对js闭包。面向对象、继承的理解**

1）闭包理解：

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

2） 面向对象：

<http://www.ruanyifeng.com/blog/2010/05/object-oriented_javascript_encapsulation.html>

3）继承：<https://segmentfault.com/a/1190000002440502>

<http://blog.csdn.net/james521314/article/details/8645815>

1. **你做的页面在哪些浏览器测试过？这些浏览器的内核分别是**

**什么？**

IE内核浏览器：360，傲游，搜狗，世界之窗，腾讯TT。

非IE内核浏览器：firefox opera safari chrome 。

IE浏览器的内核Trident、Mozilla的Gecko、Chrome的Blink（WebKit的分支）、Opera内核原为Presto，现为Blink；

1. **写出几种IE6 bug的解决方法**

1）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.

2）IE6双倍边距bug：在该元素中加入display:inline 或 display:block  
 3）像素问题 使用多个float和注释引起的 使用dislpay:inline -3px    
 4）超链接hover 点击后失效  使用正确的书写顺序 link visited hover active  
 5）z-index问题 给父级添加position:relative  
 6）Min-height 最小高度 ！Important 解决’ 7.select 在ie6下遮盖 使用iframe嵌套  
 7）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. **清楚浮动的几种放回，各自的优缺点**

**1、父级div定义伪类:after和zoom**

<style type="text/css">

 　  .div1{background:#000080;border:1px solid red;}

 　  .div2{background:#800080;border:1px solid red;height:100px;margin-top:10px}

 　  .left{float:left;width:20%;height:200px;background:#DDD}

 　  .right{float:right;width:30%;height:80px;background:#DDD}

　   /\*清除浮动代码\*/

 　  .clearfloat:after{display:block;clear:both;content:"";visibility:hidden;height:0}

 　  .clearfloat{zoom:1}

</style>

<div class="div1 clearfloat">

　　<div class="left">Left</div>

　　<div class="right">Right</div>

</div>

<div class="div2">

     div2

</div>

原理：IE8以上和非IE浏览器才支持:after，原理和方法2有点类似，zoom(IE转有属性)可解决ie6,ie7浮动问题。

优点：浏览器支持好，不容易出现怪问题（目前：大型网站都有使用，如：腾迅，网易，新浪等等）。

缺点：代码多，不少初学者不理解原理，要两句代码结合使用，才能让主流浏览器都支持。

建议：推荐使用，建议定义公共类，以减少CSS代码。

**2、父级div定义overflow:hidden**

<style type="text/css">

 　  .div1{background:#000080;border:1px solid red;

                  /\*解决代码\*/width:98%;overflow:hidden}

 　  .div2{background:#800080;border:1px solid red;height:100px;margin-                                          top:10px;width:98%}

 　  .left{float:left;width:20%;height:200px;background:#DDD}

 　  .right{float:right;width:30%;height:80px;background:#DDD}

</style>

<div class="div1">

　　<div class="left">Left</div>

　　<div class="right">Right</div>

</div>

<div class="div2">

　　div2

</div>

原理：必须定义width或zoom:1，同时不能定义height，使用overflow:hidden时，浏览器会自动检查浮动区域的高度。

优点：简单，代码少，浏览器支持好。

缺点：不能和position配合使用，因为超出的尺寸的会被隐藏。

建议：只推荐没有使用position或对overflow:hidden理解比较深的朋友使用。

**3、结尾处加空div标签clear:both**

<style type="text/css">

　   .div1{background:#000080;border:1px solid red}

　   .div2{background:#800080;border:1px solid red;height:100px;margin-top:10px}

 　  .left{float:left;width:20%;height:200px;background:#DDD}

　   .right{float:right;width:30%;height:80px;background:#DDD}

　   /\*清除浮动代码\*/

　   .clearfloat{clear:both}

</style>

<div class="div1">

　　<div class="left">Left</div>

　　<div class="right">Right</div>

　　<div class="clearfloat"></div>

</div>

<div class="div2">

　　div2

</div>

原理：添加一个空div，利用css提高的clear:both清除浮动，让父级div能自动获取到高度。

优点：简单，代码少，浏览器支持好，不容易出现怪问题。

缺点：不少初学者不理解原理；如果页面浮动布局多，就要增加很多空div，让人感觉很不爽。

建议：不推荐使用，但此方法是以前主要使用的一种清除浮动方法。

**4、父级div定义height**

<style type="text/css">

　　   .div1{background:#000080;border:1px solid red;/\*解决代码\*/height:200px;}

　　   .div2{background:#800080;border:1px solid red;height:100px;margin-top:10px}

 　　  .left{float:left;width:20%;height:200px;background:#DDD}

　　   .right{float:right;width:30%;height:80px;background:#DDD}

</style>

<div class="div1">

　　<div class="left">Left</div>

　　<div class="right">Right</div>

</div>

<div class="div2">

　　div2

</div>

原理：父级div手动定义height，就解决了父级div无法自动获取到高度的问题。

优点：简单，代码少，容易掌握。

缺点：只适合高度固定的布局，要给出精确的高度，如果高度和父级div不一样时，会产生问题。

建议：不推荐使用，只建议高度固定的布局时使用。

**5、父级div定义overflow:auto**

<style type="text/css">

 　　  .div1{background:#000080;border:1px solid red;

         /\*解决代码\*/width:98%;overflow:auto}

 　.div2{background:#800080; border:1px solid red; height:100px; margin-top:10px;width:98%}

 　　  .left{float:left;width:20%;height:200px;background:#DDD}

　　   .right{float:right;width:30%;height:80px;background:#DDD}

</style>

<div class="div1">

　　<div class="left">Left</div>

　　<div class="right">Right</div>

</div>

<div class="div2">

　　div2

</div>

原理：必须定义width或zoom:1，同时不能定义height，使用overflow:auto时，浏览器会自动检查浮动区域的高度。

优点：简单，代码少，浏览器支持好。

缺点：内部宽高超过父级div时，会出现滚动条。

建议：不推荐使用，如果你需要出现滚动条或者确保你的代码不会出现滚动条就使用吧。

**69.Javascript的typeof返回哪些数据类型；列举3种强制类型转换和2中隐式类型转换**

**1）返回数据类型**

undefined

string

boolean

number

[symbol](https://developer.mozilla.org/en-US/docs/Glossary/Symbol)(ES6)

Object

Function

1. **强制类型转换**

Number(参数)  把任何类型转换成数值类型。

parseInt(参数1，参数2)  将字符串转换成整数

parseFloat()将字符串转换成浮点数字

string(参数)：可以将任何类型转换成字符串

Boolean()  可以将任何类型的值转换成布尔值。

1. **隐式类型转换**

1.四则运算

加法运算符+是双目运算符，只要其中一个是String类型，表达式的值便是一个String。

对于其他的四则运算，只有其中一个是Number类型，表达式的值便是一个Number。

对于非法字符的情况通常会返回NaN：

'1' \* 'a' // => NaN，这是因为parseInt(a)值为NaN，1 \* NaN 还是 NaN

2.判断语句

判断语句中的判断条件需要是Boolean类型，所以条件表达式会被隐式转换为Boolean。 其转换规则同Boolean的构造函数。比如：

var obj = {};if(obj){

while(obj);}

3.Native代码调用

JavaScript宿主环境都会提供大量的对象，它们往往不少通过JavaScript来实现的。 JavaScript给这些函数传入的参数也会进行隐式转换。例如BOM提供的alert方法接受String类型的参数：

alert({a: 1}); // => [object Object]

1. **写出3个使用this的典型应用**

function Thing() { }

Thing.prototype.foo = "bar";

Thing.prototype.logFoo = function () {

console.log(this.foo);

}

Thing.prototype.setFoo = function (newFoo) {

this.foo = newFoo;

}

var thing1 = new Thing();

var thing2 = new Thing();

thing1.logFoo(); //logs "bar"

thing2.logFoo(); //logs "bar"

thing1.setFoo("foo");

thing1.logFoo(); //logs "foo";

thing2.logFoo(); //logs "bar";

thing2.foo = "foobar";

thing1.logFoo(); //logs "foo";

thing2.logFoo(); //logs "foobar";

2.

function Thing1() { }

Thing1.prototype.foo = "bar";

function Thing2() {

this.foo = "foo";

}

Thing2.prototype = new Thing1();

function Thing3() {}

Thing3.prototype = new Thing2();

var thing = new Thing3();

console.log(thing.foo); //logs "foo"

3.

function Thing() {}

Thing.prototype.foo = "bar";

Thing.prototype.logFoo = function () {

function doIt() {

onsole.log(this.foo);

}

doIt.apply(this);

}

function doItIndirectly(method) {

method();

}

var thing = new Thing();

doItIndirectly(thing.logFoo.bind(thing)); //logs bar

**对前端界面工程师这个职位是怎么样理解的？它的前景会怎**

**样？**

前端工程师属于一个比较新兴的技术，各种技术层出不穷，随着客户体验的重要性前端需要掌握的技能也越来越多了，对前端的要求也越来越多了，而且我们离客户很近，除了掌握必要的技能还要掌握用户的心理，属于沟通。 前景：前景无疑是值得肯定的，也需要我们时刻关注最新的技术，这会是一个时刻都在学习的道路

1. **JQuery delegate函数的作用？请举例说明**

delegate() 方法为指定的元素（属于被选元素的子元素）添加一个或多个事件处理程序，并规定当这些事件发生时运行的函数

例如点击div中的p标签 让 弹个窗口

$(‘div’).delegate(‘p’, ‘click’, function(){ alert();}

1. **Eval函数的作用**

eval可以将字符串生成语句执行，一般执行动态的js语句。   
eval的使用场合：有时候我们预先不知道要执行什么语句，只有当条件和参数给时才知道执行什么语句，这时候eval就派上用场了。

1. **标签上title与alt属性的区别是什么**

title 是鼠标放上去的额外信息 alt 是不能正常显示的信息

1. **对WEB标准以及w3c的理解与认识？**

Web标准就是将页面的解构、表现和行为各自独立实现，w3c对标注提出了规范化的要求1.对结构的要求：（标签规范可以提高搜索引擎对页面的抓取效率，对SEO很有帮助）

1）标签字母要小写；

2）标签要闭合；

3）标签不允许随意嵌套。

2.对css和js的要求：

1）尽量使用外联css样式表和js脚本，使结构、表现和行为分成三块，符合规范，同时提高页面渲染速度，提高用户体验；

2）样式尽量少用行间样式表，使结构与表现分离，标签的id和class命名要做到见文知义，标签越少，加载越快，用户体验更高，代码维护更简单，便于改版；

3）不需要变动页面内容，便可提供打印版本而不需要复制内容，提高网站易用性

1. **Css选择符有哪些？哪些属性可以继承？优先级算法如何计**

**算？**

ID 和 CLASS ；Class 可继承 ；伪类A标签可以继承；列表 UL LI DL DD DT 可继承

；优先级就近原则，样式定义最近者为准；载入样式以最后载入的定位为准

优先级为

!important > [ id > class > tag ]

Important 比 内联优先级高

1. **请戳出ie6/7下特有的cssbug**

**一：li边距“无故” 增加**

设置ul的显示形式为\*display:inline-block;即可，前面加\*是只 针对IE6/IE7有效

**二：IE6 不支持min-height属性，但它却认为height就是最小高度**

使用ie6不支持但其余浏览器支持的属性!important。

**三:Overflow:**

在IE6/7中，overflow无法正确的隐藏有相对定位position:relative;的子元素。解决方法就是给外包容器.wrap加上position:relative;。

**四：border：none 在IE6不起作用： 写成border：0 就可以了，**

**五:100%高度**

在IE6下，如果要给元素定义100%高度，必须要明确定义它的父级元素的高度，如果你需要给元素定义满屏的高度，就得先给html和body定义 height:100%;。

**六:双边距 Bug**

当元素浮动时，IE6会错误的的把浮动方式的margin值双倍计算，给float的元素添加一个display：inline

**七：躲猫猫bug**

些定义了:hover的链接，当鼠标移到那些链接上时，在IE6下就会触发躲猫猫。  
1.在（那个未浮动的）内容之后添加一个<span style=”clear: both;”> </span>  
2.触发包含了这些链接的容器的hasLayout，一个简单的方法就是给其定义height:1%;

**八:IE6 绝对定位的元素1px 间距bug**

当绝对定位的父元素或宽度为奇数时，bottom和right会多出现1px，

解决方案，针对IE6进行hack处理

1. **如何将一个元素600毫秒的速度缓慢向上滑动显示？**

如果需要在父元素底部向上，可以利用margin-top 把子元素，挤下去，同事父元素设置隐藏，然后改变margintop的值也可以利用定来做，把子元素定位最下边

(function(){

var oDiv = document.createElement('div');

oDiv.style.width = '100px';

oDiv.style.height = '100px';

oDiv.style.backgroundColor = 'red';

oDiv.style.position = 'absolute';

oDiv.style.marginTop = 100 + 'px';

document.body.appendChild(oDiv);

var timer = setInterval(function(){

var m = parseInt(oDiv.style.marginTop);

if (m == 0 ) {

clearInterval(timer);

return;

}

oDiv.style.marginTop = parseInt(oDiv.style.marginTop) - 1 + 'px';

},600);

})();

1. **写一个获取非行间样式的函数**

Function getStyle(obj, attr){

If(obj.currentStyle){

return obj.currentStyle[attr];

}else{

return getComputedStyle(obj,false)[attr];

}

}

1. **请用正则表达式验证数字**

^[0-9]\*$

1. **为什么利用多个域名来提供网站资源会更有效？**
2. 突破浏览器的并发限制（浏览器同一域名最大的并发请求数量为6个，ie6为2个）
3. 节约cookie带宽
4. CDN缓存更方便
5. 防止不必要的安全问题（尤其是cookie的隔离尤为重要）
6. 节约主机域名连接数，优化页面响应速度
7. **你如何从浏览器的URL中获取参数信息**

浏览器宿主环境中，有一个location对象，同时这个对象也是window对象和document对象的属性。

location对象中提供了与当前窗口加载的文档有关的的信息，即URL信息。

如 https://www.baidu.com/api/sousu?search=baidu&id=123#2

location.href: 完整URL

location.protocol: 返回协议（https:）

location.host: 返回服务器名称和端口号（[www.baidu.com](http://www.baidu.com)）

location.hostname: 返回服务器名称（[www.baidu.com](http://www.baidu.com)）

location.port:返回服务器端口号（http默认80，https默认443）

location.pathname:返回URL中的目录和文件名（api/sousu）

location.search:返回查询字符串（?search=baidu&id=123#2）

location.hash:返回hash值（#2）

1. **手机端文字大小用什么单位**

对于只需要适配少部分手机设备，且分辨率对页面影响不大的，使用px即可

对于需要适配各种移动设备，使用rem，例如只需要适配iPhone和iPad等分辨率差别比较挺大的设备

1. **是否做过有上百图层的psd切图？ps隐藏其他图层，只显示其中一个图层的快捷键**

Alt + 当前图层前眼睛

1. **浏览器标准模式和怀疑模式之间的区别是什么？**

这是个历史遗留问题，W3C标准推出前，旧的页面都是根据旧的渲染方式对页面进行渲染的，因此在W3C标准推出后为了保证旧页面的正常显示，保持浏览器的兼容性，这样浏览器上就产生了能够兼容W3C标准渲染的严格模式和保证旧页面显示的怪异模式的标准兼容模式。

具体表现：

1.在严格模式中 ：width是内容宽度 ，元素真正的宽度 = margin-left + border-left-width + padding-left + width + padding-right + border-right- width +  margin-right;

在怪异模式中 ：width则是元素的实际宽度 ，内容宽度 = width - ( padding-left + padding-right + border-left-width + border-right-width)

2）可以设置行内元素的高宽

    在标准模式下，给span等行内元素设置wdith和height都不会生效，而在怪异模式下，则会生效。

3）可设置百分比的高度

    在标准模式下，一个元素的高度是由其包含的内容来决定的，如果父元素没有设置高度，子元素设置一个百分比的高度是无效的。

4）用margin:0 auto设置水平居中在IE下会失效

    使用margin:0 auto在标准模式下可以使元素水平居中，但在怪异模式下却会失效, 怪异模式下的解决办法，用text-align属性:

   body{text-align:center};#content{text-align:left}

5）怪异模式下设置图片的padding会失效

6）怪异模式下Table中的字体属性不能继承上层的设置

7）怪异模式下white-space:pre会失效

1. **Javascript的同源策略**

同源策略是Javascript重要的安全度量标准。它最早出自Netscape Navigator2.0，其目的是防止某个文档或脚本从多个不同源装载。所谓的同源就是同协议，同主机名，同端口号。

它的精髓很简单：它认为自任何站点装载的信赖内容是不安全的。当被浏览器半信半疑的脚本运行在沙箱时，它们应该只被允许访问来自同一站点的资源，而不是那些来自其它站点可能怀有恶意的资源。

1. **你所做的WEB移动端项目中，有遇到哪些兼容?**

给大家推荐个网址：

http://www.jb51.net/article/84973.htm

1. **有了解响应式布局吗？请大体说一说**

响应式布局概念：Responsive design，意在实现不同屏幕分辨率的终端上浏览网页的不同展示方式。通过响应式设计能使网站在手机和平板电脑上有更好的浏览阅读体验。

设计步骤：

1. 设置meta标签
2. 根据媒体查询设置样式
3. 设置多种视图宽度

注意点：

1. 宽度使用百分比
2. 处理图片缩放问题
3. **身为以为web前端工程师，你肯定知道现在最流行的前端技**

**术吧，有那些？**

Vuejs2.0/Angular2.0/React Native /es6//Nodejs

http2

gulp/webpack

1. **请简述为什么要使用数据库的事务**

数据库事务(Database Transaction) ，是指作为单个逻辑工作单元执行的一系列操作，要么完全地执行，要么完全地不执行。

**原子性（Atomic）（Atomicity)**  
　　事务必须是原子工作单元；对于其数据修改，要么全都执行，要么全都不执行。通常，与某个事务关联的操作具有共同的目标，并且是相互依赖的。如果系统只执行这些操作的一个子集，则可能会破坏事务的总体目标。原子性消除了系统处理操作子集的可能性。  
　　**一致性（Consistent）(Consistency)**  
　　事务在完成时，必须使所有的数据都保持一致状态。在相关数据库中，所有规则都必须应用于事务的修改，以保持所有数据的完整性。事务结束时，所有的内部数据结构（如 B 树索引或双向链表）都必须是正确的。某些维护一致性的责任由应用程序开发人员承担，他们必须确保应用程序已强制所有已知的完整性约束。例如，当开发用于转帐的应用程序时，应避免在转帐过程中任意移动小数点。  
　　**隔离性（Insulation）(Isolation)**  
　　由并发事务所作的修改必须与任何其它并发事务所作的修改隔离。事务查看数据时数据所处的状态，要么是另一并发事务修改它之前的状态，要么是另一事务修改它之后的状态，事务不会查看中间状态的数据。这称为隔离性，因为它能够重新装载起始数据，并且重播一系列事务，以使数据结束时的状态与原始事务执行的状态相同。当事务可序列化时将获得最高的隔离级别。在此级别上，从一组可并行执行的事务获得的结果与通过连续运行每个事务所获得的结果相同。由于高度隔离会限制可并行执行的事务数，所以一些应用程序降低隔离级别以换取更大的吞吐量。  
　**持久性（Duration）(Durability）**  
　　事务完成之后，它对于系统的影响是永久性的。该修改即使出现致命的系统故障也将一直保持。