**第一阶段**

1. doctype的作用。

<!DOCTYPE> 声明位于文档中的最前面的位置，处于 <html> 标签之前。此标签可告知浏览器文档使用哪种 HTML 或 XHTML 规范该标签可声明三种 DTD 类型，分别表示严格版本、过渡版本以及基于框架的 HTML 文档。

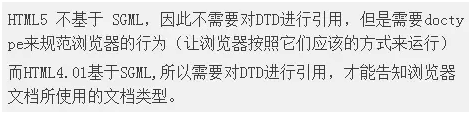
1. 介绍一下CSS的盒子模型？

第一种是W3C的标准模型，另一种是IE的传统模型  
W3C盒子由margin，border，padding，content  
IE盒子由margin，padding，padding，content构成，但是由于content包括padding，border，即为margin+content

3、Doctype作用？标准模式与兼容模式各有什么区别?

标准模式的排版 和JS运作模式都是以该浏览器支持的最高标准运行。在兼容模式中，页面以宽松的向后兼容的方式显示,模拟老式浏览器的行为以防止站点无法工作。

4、HTML5 为什么只需要写 <!DOCTYPE HTML>？



5、行内元素有哪些？块级元素有哪些？ 空(void)元素有那些？

行内Span a I em b 块级 div p br ul ol dl

空元素：即系没有内容的HTML元素，例如：br、meta、hr、link、input、img

6、页面导入样式时，使用link和@import有什么区别？

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  
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。

7、介绍一下你对浏览器内核的理解？

Trident，这个浏览器内核是微软IE的排版引擎的名称，首次被应用于IE4。这个引擎被设计成一个软件组件，使其他软件开发人员很容易将网页浏览功能添加到他们自行开发的应用程序里，它的缺点是其对真正的网页标准支持不是很好，同事存在许多bug。它也有一定的局限性，那就是它只能被应用于Windows平台。

Gecko，它是一套由自由型开放源代码、以c++编写的排版引擎，它原本由网景通讯公司开发，现在由Mozilla基金会维护，它也是最流行的排版引擎之一，流行程度仅次于Trident。它提供了一个丰富的应用程序接口以供互联网相关的应用程序使用，其缺点是消耗内存资源多。它不像Trident那样局限，它可以跨平台在Windows、**[Linux](http://lib.csdn.net/base/linux" \t "_blank" \o "Linux知识库)**、MacOS上都可以使用。

  Presto，它是由Opera Software开发的网页浏览器排版引擎，它实际上是一个动态内核，它的特点是渲染速度的优化达到了极致，它是目前公认网页浏览速度最快的浏览器内核，而代价是牺牲了网页的兼容性。它的不足之处是它是一个商业引擎，在很大程度上限制了它的发展。它同样可以跨平台使用。

  最后就是Webkit了，它是一个开放源代码的浏览器引擎，它是一种用来让网页浏览器绘制网页的排版引擎，被用于苹果的Safari。其优点是有清晰的源码结构，渲染速度快，缺点是兼容性不好。

8、常见的浏览器内核有哪些？

答：Chromium内核，以Chrome为代表  
Gecko内核，以Firefox为代表-moz-  
Presto内核，以老版本的Opera为代表-o-  
Webkit内核，以Sarfari为代表-webkit-  
Trident内核，以IE为代表 -ms-

9、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

语义header和footer  
以往的网页结构中我们习惯给网页头部和底部 通过id或者header和footerh5中直接语义话了article元素表示页面中一块与上下文不相关的独立内容。比如一篇文章。aside元素表示article元素内容之外的、与article元素内容相关的辅助信息。header元素表示页面中一个内容区块或真个页面的标题footer元素表示整个页面或页面中一个内容区块的脚注。一般来说，他会包含创作者的姓名、创作日期以及创作者的联系信息。

移除的元素

废除listing使用pre替代  
废除xmp使用code替代  
废除nextid使用guids  
废除plaintex使用“text/plian”（无格式正文）MIME类型替代。

如何处理兼容

1.IE8/IE7/IE6支持通过document.方法产生的标签，利用这一特性让这些浏览器支持HTML5新标签。  
2.使用是html5shim框架  
另外，DOCTYPE声明的方式是区分HTML和HTML5标志的一个重要因素，此外，还可以根据新增的结构、功能元素来加以区分。

10、简述一下你对HTML语义化的理解？

1为了在没有CSS的情况下，页面也能呈现出很好地内容结构、代码结构:为了裸奔时好看；  
2用户体验：例如title、alt用于解释名词或解释图片信息、label标签的活用；  
3有利于SEO：和搜索引擎建立良好沟通，有助于爬虫抓取更多的有效信息：爬虫依赖于标签来确定上下文和各个关键字的权重；  
4方便其他设备解析（如屏幕阅读器、盲人阅读器、移动设备）以意义的方式来渲染网页；  
5便于团队开发和维护，语义化更具可读性，是下一步吧网页的重要动向，遵循W3C标准的团队都遵循这个标准，可以减少差异化。

11、iframe有那些缺点？

即时内容为空，加载也需要时间  
会阻止页面加载  
没有语意

12、列出display的值，说明他们的作用。position的值， relative和absolute定位原点是？

Display：none/block/inline/inline-block

none 隐藏

block 块显示

inline 内嵌

Inline-block 行内块

table 表格显示

list-item 项目列表

1.position:relative; 如果对一个元素进行相对定位，首先它将出现在它所在的位置上。然后通过设置垂直或水平位置，让这个元素"相对于"它的原始起点进行移动。（再一点，相对定位时，无论是否进行移动，元素仍然占据原来的空间。因此，移动元素会导致它覆盖其他框）

相对定位：relative 没有脱离正常的文档流，被设置元素相对于其原始位置而进行定位，其原始占位信息仍存在

2.position:absolute; 表示绝对定位，位置将依据浏览器左上角开始计算。 绝对定位使元素脱离文档流，因此不占据空间。普通文档流中元素的布局就像绝对定位的元素不存在时一样。（因为绝对定位的框与文档流无关，所以它们可以覆盖页面上的其他元素并可以通过z-index来控制它层级次序。z-index的值越高，它显示的越在上层。）

绝对定位：ablution 脱离了文档流与浮动模型，独立于其他对象而存在，没有占位。

3.父容器使用相对定位，子元素使用绝对定位后，这样子元素的位置不再相对于浏览器左上角，而是相对于父窗口左上角

13、一个满屏 品 字布局 如何设计?

14、常见兼容性问题？

浏览器兼容问题一：不同浏览器的标签默认的margin和padding不同  
浏览器兼容问题二：块属性标签float后，又有横行的margin情况下，在IE6显示margin比设置的大  
浏览器兼容问题三：设置较小高度标签（一般小于10px），在IE6，IE7，遨游中高度超出自己设置高度  
浏览器兼容问题四：行内属性标签，设置display:block后采用float布局，又有横行的margin的情况，IE6间距bug  
浏览器兼容问题五：图片默认有间距  
浏览器兼容问题六：标签最低高度设置min-height不兼容  
浏览器兼容问题七：各种特殊样式的兼容，比如透明度、圆角、阴影等。特殊样式每个浏览器的代码区别很大，所以，只能现查资料通过给不同浏览器写不同的代码来解决。

15、如果需要手动写动画，你认为最小时间间隔是多久，为什么？（阿里）

30毫秒 比较流畅！

16、列举IE 与其他浏览器不一样的特性？

a. IE的排版引擎是Trident （又称为MSHTML）  
b. Trident内核曾经几乎与W3C标准脱节（2005年）  
c. Trident内核的大量 Bug等安全性问题没有得到及时解决  
d. JS方面，有很多独立的方法，例如绑定事件的attachEvent、创建事件的createEventObject等  
e. CSS方面，也有自己独有的处理方式，例如设置透明，低版本IE中使用滤镜的方式

17、99%的网站都需要被重构是那本书上写的？

应用web标准进行设计（第2版）

26、css引入的方式有几种？ link和import的区别是什么？

链入外部样式表、内部样式表、导入外部样式表,用@import、内嵌样式  
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  
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

27、前端页面有哪三层构成？ 分别是什么？ 作用是什么？

HTML结构层 运用HTML搭建页面框架

Css样式层 显示内容的样式

Js行为层 对用户的操作进行处理

29、如何用css分别单独定义ie6 IE7 IE8 IE9 IE10的width属性

仅IE6识别 : \_margin-left:10px;  
IE6/IE7识别 ： \*margin-left:10px;  
仅IE8识别 ： margin-left:10px\0;  
仅IE识别 ： margin-left:10px\9;

30、在css中哪些属性可以同父元素继承。

所有元素可继承：visibility和cursor。  
内联元素可继承：letter-spacing、word-spacing、white-space、line-height、color、font、font-family、font-size、font-style、font-variant、font-weight、text-decoration、text-transform、direction。  
终端块状元素可继承：text-indent和text-align。  
列表元素可继承：list-style、list-style-type、list-style-position、list-style-image。

31、谈谈以前端的角度出发做好seo需要做什么

 精减代码   CSS Sprites 为图片指定宽度与高度  启用Keep-Alive属性  使用浏览器缓存

32、css选择符有哪些？哪些属性可以继属性承？优先级算法如何计算？css3新增的

CSS 元素选择器 CSS 类选择器 CSS ID 选择器 CSS 属性选择器 CSS 后代选择器 CSS 子元素选择器 CSS 相邻兄弟选择器 CSS 伪类 CSS 伪元素css通配符选择器  
CSS3中新增了3个选择器：  
E[attr^="value"]——属性值以value开头  
E[attr$="value"]——属性值以value结尾  
E[attr\*="value"]——属性值包含value  
所有元素可继承：visibility和cursor。  
内联元素可继承：letter-spacing、word-spacing、white-space、line-height、color、font、font-family、font-size、font-style、font-variant、font-weight、text-decoration、text-transform、direction。  
终端块状元素可继承：text-indent和text-align。  
列表元素可继承：list-style、list-style-type、list-style-position、list-style-image。  
算法：  
在多个选择符应用于同一个元素上那么Specificity值高的最终获得优先级。  
选择符Specificity值列表：规则：  
1. 行内样式优先级Specificity值为1,0,0,0，高于外部定义。  
如：sjweb  
外部定义指经由或标签定义的规则；  
2.!important声明的Specificity值最高；  
3.Specificity值一样的情况下，按CSS代码中出现的顺序决定，后者CSS样式居上；  
4.由继续而得到的样式没有specificity的计算，它低于一切其他规则(比如全局选择符\*定义的规则)。  
算法：当遇到多个选择符同时出现时候  
按选择符得到的Specificity值逐位相加，  
{数位之间没有进制 比如说： 0,0,0,5 + 0,0,0,5 =0,0,0,10 而不是 0,0, 1, 0}  
就得到最终计算得的specificity，  
然后在比较取舍时按照从左到右的顺序逐位比较。

33、移动端（Android IOS）怎么做好用户体验?

**一.明确、重点突出的内容**

## 二.菜单和导航：保持简单化

## 三.流式的布局

## 四.触屏设计

## 五.表单最小化

## 六.放弃图片

## 七.充分利用手机的特定功能

34、你在现在的团队处于什么样的角色，起到了什么明显的作用？

35、你认为怎样才是全端工程师（Full Stack developer）？

36、css基本语句的构成

（自定义的样式名称）｛

样式内容

｝

例:

li(为li标签设定){

width:300px;<--!宽300-->

height:22px;<--!高22-->

font:12px/23px 宋体;<--!字体大小/行高 字体样式-->

text-align:center;<--!对其方式:居中-->

background:url(on.gif) no-repeat 45px -80px;<--! 背景图片 平铺样式-->

list-style:none;<--! li标签显示方式:无提示标记-->

}

37、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)

38、什么是html5

万维网的核心语言、标准通用标记语言下的一个应用超文本标记语言（HTML）的第五次重大修改

39、哪些浏览器支持html5

Chrome 15.0.874，IE 9.0.8112，Firefox 8.0.1，Opera 11.52 （国内）[360浏览器](https://www.baidu.com/s?wd=360%E6%B5%8F%E8%A7%88%E5%99%A8&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YzPhNWP1nkP1fvmWwWmWck0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnHc1rjb4P1cYn1Ddn10Lnjmdrf" \t "_blank) 4.0.3.8，[搜狗浏览器](https://www.baidu.com/s?wd=%E6%90%9C%E7%8B%97%E6%B5%8F%E8%A7%88%E5%99%A8&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YzPhNWP1nkP1fvmWwWmWck0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnHc1rjb4P1cYn1Ddn10Lnjmdrf" \t "_blank) 3.1.0.3688，[遨游浏览器](https://www.baidu.com/s?wd=%E9%81%A8%E6%B8%B8%E6%B5%8F%E8%A7%88%E5%99%A8&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YzPhNWP1nkP1fvmWwWmWck0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnHc1rjb4P1cYn1Ddn10Lnjmdrf" \t "_blank)v3.2.2.1000，[QQ浏览器](https://www.baidu.com/s?wd=QQ%E6%B5%8F%E8%A7%88%E5%99%A8&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YzPhNWP1nkP1fvmWwWmWck0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnHc1rjb4P1cYn1Ddn10Lnjmdrf" \t "_blank) 6.8(10793)

40、css 中的使用列布局是什么？ 为了实现列布局我们需要指定哪些内容？

flex布局；在使用时，通过 CSS 属性声明首先定义主轴的方向（水平或垂直），则交叉轴的方向也相应确定下来

41、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.

42、写出5 种以上ie6 bug 的解决方法，哪些你认为是解决起来最麻烦的

CSS 元素选择器 CSS 类选择器 CSS ID 选择器 CSS 属性选择器 CSS 后代选择器 CSS 子元素选择器 CSS 相邻兄弟选择器 CSS 伪类 CSS 伪元素css通配符选择器  
CSS3中新增了3个选择器：  
E[attr^="value"]——属性值以value开头  
E[attr$="value"]——属性值以value结尾  
E[attr\*="value"]——属性值包含value  
所有元素可继承：visibility和cursor。  
内联元素可继承：letter-spacing、word-spacing、white-space、line-height、color、font、font-family、font-size、font-style、font-variant、font-weight、text-decoration、text-transform、direction。  
终端块状元素可继承：text-indent和text-align。  
列表元素可继承：list-style、list-style-type、list-style-position、list-style-image。  
算法：  
在多个选择符应用于同一个元素上那么Specificity值高的最终获得优先级。  
选择符Specificity值列表：规则：  
1. 行内样式优先级Specificity值为1,0,0,0，高于外部定义。  
如：sjweb  
外部定义指经由或标签定义的规则；  
2.!important声明的Specificity值最高；  
3.Specificity值一样的情况下，按CSS代码中出现的顺序决定，后者CSS样式居上；  
4.由继续而得到的样式没有specificity的计算，它低于一切其他规则(比如全局选择符\*定义的规则)。  
算法：当遇到多个选择符同时出现时候  
按选择符得到的Specificity值逐位相加，  
{数位之间没有进制 比如说： 0,0,0,5 + 0,0,0,5 =0,0,0,10 而不是 0,0, 1, 0}  
就得到最终计算得的specificity，  
然后在比较取舍时按照从左到右的顺序逐位比较。  
  
  
  
（自定义的样式名称）｛  
    样式内容  
｝  
例:  
li(为li标签设定){  
width:300px;<--!宽300-->  
height:22px;<--!高22-->  
font:12px/23px 宋体;<--!字体大小/行高   字体样式-->  
text-align:center;<--!对其方式:居中-->  
background:url(on.gif) no-repeat 45px -80px;<--! 背景图片  平铺样式-->  
list-style:none;<--! li标签显示方式:无提示标记-->  
}  
  
  
  
万维网的核心语言、标准通用标记语言下的一个应用超文本标记语言（HTML）的第五次重大修改  
  
  
flex布局；在使用时，通过 CSS 属性声明首先定义主轴的方向（水平或垂直），则交叉轴的方向也相应确定下来  
  
  
  
1、IE6怪异解析之padding与border算入宽高   
原因：未加文档声明造成非盒模型解析   
解决方法：加入文档声明<!doctype html>   
  
2、IE6在块元素、左右浮动、设定marin时造成margin双倍（双边距）   
解决方法：display:inline   
  
3、以下三种其实是同一种bug，其实也不算是个bug，举个例子：父标签高度20，子标签11，垂直居中，20-11=9，9要分给文字的上面与下面，怎么分？IE6就会与其它的不同，所以，尽量避免。   
1）字体大小为奇数之边框高度少1px   
解决方法：字体大小设置为偶数或line-height为偶数   
2）line-height，文本垂直居中差1px   
解决方法：padding-top代替line-height居中，或line-height加1或减1   
3）与父标签的宽度的奇偶不同的居中造成1px的偏离   
解决方法：如果父标签是奇数宽度，则子标签也用奇数宽度;如果是父标签偶数宽度，则子标签也用偶数宽度   
  
4、内部盒模型超出父级时，父级被撑大   
解决方法：父标签使用overflow:hidden   
  
5、line-height默认行高bug   
解决方法：line-height设值   
  
6、行标签之间会有一小段空白   
解决方法：float或结构并排(可读性差，不建议)   
  
7、标签高度无法小于19px   
解决方法：overflow: hidden;   
  
8、左浮元素margin-bottom失效   
解决方法：显示设置高度 or 父标签设置\_padding-bottom代替子标签的margin-bottom or 再放个标签让父标签浮动，子标签   
margin- bottom，即(margin-bottom与float不同时作用于一个标签)   
  
9、img于块元素中，底边多出空白   
解决方法：父级设置overflow: hidden; 或 img { display: block; } 或 \_margin: -5px;   
  
10、li之间会有间距   
解决方法：float: left;

43、哪些浏览器支持html5？这些浏览器的内核分别是什么

谷歌 Webkit 火狐Mozilla ie trident Opera  Presto Safari webkit

44、什么是Web workers？为什么我们需要他

Web Workers为WEB前端网页上的脚本提供了一种能在后台进程中运行的方法。一旦它被创建，Web Workers就可以通过postMessage()向任务池发送任务请求，执行完之后再通过postMessage()返回消息给创建者指定的事件处理程序(通过onmessage进行捕获)。  
Web Workers进程能够在不影响用户界面的情况下处理任务，并且，它还可以使用XMLHttpRequest来处理I/O，无论responseXML和channel属性是否为null。

45、前端页面有那三部分构成，其作用是什么。

HTML结构层 运用HTML搭建页面框架

Css样式层 显示内容的样式

Js行为层 对用户的操作进行处理

46、写出至少三个css3中新添加的样式属性的名字及参数。

border-radius：允许向元素添加圆角border-radius:5px 10px 15px 20px;  
box-shadow：阴影box-shadow:10px 10px 5px 5px gray;/\* 阴影\*/  
box-shadow:10px 10px 5px 5px inset;/\* 阴影 inset内阴影\*/  
 文字加倒影：font-size:36px; -webkit-box-reflect:below -10px;  
 -webkit-linear-gradient:(top,transparent 50%,#FFFFFF);  
  
  3D旋转.div1{  
            transform:rotateX(120deg);  
            transform:rotateY(120deg);  
            transform:rotateZ(120deg);  
        }  
                   
  
  过渡属性 div{  
            width:200px;  
            height:200px;  
            border:1px solid red;  
            background:blue;  
            transition:width 2s,background 2s;/\* 过渡的属性与时间\*/  
        }  
        div:hover{  
            width:600px;  
            background:yellow;  
        }

47、给一个元素添加css样式有哪几种方法说说他们的优先级。

行内样式 内部样式 外部样式表 行内<内部<外部

48、多人项目中，你如何规划css文件样式命名。

全局样式：global.css;  
框架布局：layout.css;  
字体样式：font.css;  
链接样式：link.css;  
打印样式：print.css;  
用字母、\_号工、-号、数字组成，必须以字母开头，不能为纯数字。为了开发后样式名管理方便，大家请用有意义的单词或缩写组合来命名，让同事一看就明白这样式大概是哪一块的，这样就节省了查找样式的时间

49、为了让网页更快的加载，你会如何处理你的css文件和js文件，以及图片文件，页面性能优化你还知道哪些方法。

 技术优化 代码优化 目录结构优化 针对搜索引擎的优化 人文优化

50、你经常遇到的浏览器兼容性有哪些？通常是怎么处理的。

浏览器兼容问题一：不同浏览器的标签默认的外补丁和内补丁不同  
解决方案：CSS里    \*{margin:0;padding:0;}  
浏览器兼容问题二：块属性标签float后，又有横行的margin情况下，在IE6显示margin比设置的大  
问题症状:常见症状是IE6中后面的一块被顶到下一行  
解决方案：在float的标签样式控制中加入 display:inline;将其转化为行内属性  
浏览器兼容问题三：设置较小高度标签（一般小于10px），在IE6，IE7，遨游中高度超出自己设置高度  
问题症状：IE6、7和遨游里这个标签的高度不受控制，超出自己设置的高度  
解决方案：给超出高度的标签设置overflow:hidden;或者设置行高line-height 小于你设置的高度。  
浏览器兼容问题四：行内属性标签，设置display:block后采用float布局，又有横行的margin的情况，IE6间距bug  
问题症状：IE6里的间距比超过设置的间距  
解决方案：在display:block;后面加入display:inline;display:table;  
浏览器兼容问题五：图片默认有间距  
问题症状：几个img标签放在一起的时候，有些浏览器会有默认的间距，加了问题一中提到的通配符也不起作用。  
解决方案：使用float属性为img布局  
浏览器兼容问题六：标签最低高度设置min-height不兼容  
问题症状：因为min-height本身就是一个不兼容的CSS属性，所以设置min-height时不能  
解决方案：如果我们要设置一个标签的最小高度200px，需要进行的设置为：{min-height:200px; height:auto !important; height:200px; overflow:visible;}  
浏览器兼容问题七：透明度的兼容CSS设置  
做兼容页面的方法是：每写一小段代码（布局中的一行或者一块）我们都要在不同的浏览器中看是否兼容，当然熟练到一定的程度就没这么麻烦了。建议经常会碰到兼容性问题的新手使用。很多兼容性问题都是因为浏览器对标签的默认属性解析不同造成的，只要我们稍加设置都能轻松地解决这些兼容问题。如果我们熟悉标签的默认属性的话，就能很好的理解为什么会出现兼容问题以及怎么去解决这些兼容问题。  
1./\* CSS hack\*/   
我很少使用hacker的，可能是个人习惯吧，我不喜欢写的代码IE不兼容，然后用hack来解决。不过hacker还是非常好用的。使用hacker我可以把浏览器分为3类：IE6 ；IE7和遨游；其他（IE8 chrome ff safari opera等）  
◆IE6认识的hacker 是下划线\_ 和星号 \*  
◆IE7 遨游认识的hacker是星号 \*  
比如这样一个CSS设置：  
1.height:300px;\*height:200px;\_height:100px;   
IE6浏览器在读到height:300px的时候会认为高时300px；继续往下读，他也认识\*heihgt， 所以当IE6读到\*height:200px的时候会覆盖掉前一条的相冲突设置，认为高度是200px。继续往下读，IE6还认识\_height,所以他又会覆盖掉200px高的设置，把高度设置为100px；  
IE7和遨游也是一样的从高度300px的设置往下读。当它们读到\*height200px的时候就停下了，因为它们不认识\_height。所以它们会把高度解析为200px，剩下的浏览器只认识第一个height:300px;所以他们会把高度解析为300px。因为优先级相同且想冲突的属性设置后一个会覆盖掉前一个，所以书写的次序是很重要的。

51、HTML中div与span区别；

Div 块级 span 行级

52、HTML5 存储类型有哪些，以及与他们的区别；

session级别的sessionStorage,只在当前域名内才有效，当浏览器窗口关闭，存入的值跟着也消失，它提供三个方法getItem("key")\setItem(key,value)\removeItem(key);  
cookie级别的localStorage，永久存在浏览器中，方法跟sessionStorage一样  
本地存储-谈到存储，一般说的是数据data，不管这个应用是在线还是离线。  
离线应用-这个对应的是在线，也就是，没有网的情况下，也可以使用。  
后者需要把代码、程序以及资源都放到本地，如果想要使用数据，则利用本地存储。

53、css实现垂直水平居中。

1. **用inline-block和vertical-align来实现居中**：这种方法适合于未知宽度高度的情况下。
2. **用相对绝对定位和负边距实现上下左右居中**：高度和宽度已知
3. **.使用css3中的transform来时实现水平垂直居中**：适合高度宽度未知的情况
4. **使用table-cell，inline-block实现水平垂直居中**：适合高度宽度未知的情况
5. **利用绝对定位来实现居中**：适合高度，宽度已知的情况。

54、浏览器内核分别是什么；

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

55、列举5种以上表单元素中input的type类型；

* email
* url
* number
* range
* Date pickers (date, month, week, time, datetime, datetime-local)
* search
* color

56、alt和title分别表示什么含义以及具体应用体现；

使用alt属性是为了给那些不能看到你文档中图像的浏览者提供文字说明  
使用title属性提供非本质的额外信息。大部分的可视化浏览器在鼠标悬浮在特定元素上时显示title文字为提示信息

57、css中position的属性值都有哪些？并描述其含义及具体解释；

position用来设定元素的定位类型，有absolute（[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=mv6quAkxTZn0IZRqIHckPjm4nH00T1YknvfvmHfdPHc1PhcLuHNB0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3En10YPHR1PHmL" \t "_blank)）、relative（[相对定位](https://www.baidu.com/s?wd=%E7%9B%B8%E5%AF%B9%E5%AE%9A%E4%BD%8D&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YknvfvmHfdPHc1PhcLuHNB0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3En10YPHR1PHmL" \t "_blank)）、static（静态定位，默认值）、fixed（固定定位）四种。  
static：默认。位置设置为 static 的元素，它始终会处于页面流给予的位置（static 元素会忽略任何 top、bottom、left 或 right 声明）。  
relative：位置被设置为 relative 的元素，可将其移至相对于其正常位置的地方，因此 "left:20" 会将元素移至元素正常位置左边 20 个像素的位置。  
absolute：位置设置为 absolute 的元素，可定位于相对于包含它的元素的指定坐标。此元素的位置可通过 "left"、"top"、"right" 以及 "bottom" 属性来规定。  
fixed：位置被设置为 fixed 的元素，可定位于相对于浏览器窗口的指定坐标。此元素的位置可通过 "left"、"top"、"right" 以及"bottom" 属性来规定。不论窗口滚动与否，元素都会留在那个位置。工作于 IE7（strict 模式）。

58、对html语义化理解，以及语义化有哪些优点。

1为了在没有CSS的情况下，页面也能呈现出很好地内容结构、代码结构:为了裸奔时好看；  
2用户体验：例如title、alt用于解释名词或解释图片信息、label标签的活用；  
3有利于SEO：和搜索引擎建立良好沟通，有助于爬虫抓取更多的有效信息：爬虫依赖于标签来确定上下文和各个关键字的权重；  
4方便其他设备解析（如屏幕阅读器、盲人阅读器、移动设备）以意义的方式来渲染网页；  
5便于团队开发和维护，语义化更具可读性，是下一步吧网页的重要动向，遵循W3C标准的团队都遵循这个标准，可以减少差异化。

59、经常使用的页面开发工具级测试工具。

开发工具： 测试工具：

WebStorm 谷歌自带开发者工具

HBuilder firefox

60、经常使用什么脚本库，开发或使用什么应用或组件；

用JavaScript脚本开发的轻量级UI组件；Jquery

61、使用css如何让一个宽度为200px的div水平居中，（要求兼容ie，可提供多种方法）；

62、简要画出盒模型，并描述关键要素的含义；



]9

63、页面构造中你遇到过什么样的兼容问题，如何解决

1. 同浏览器对HTML标记所具有的内外边距属性具有不同的定义。
2. 因此如果想消除这种差距，应该在相应的CSS部分加入以下CSS代码：
3. \*{margin:0px;padding:0px;}
4. 借于此，所有标记的内外边距被统一起来。
5. 优先级问题：
6. 对于同一标记属性所给定的值，有不同的优先级。其中优先级最高的是内联代码，其实是页内CSS，接下来是浏览器默认设置，最后才是外部CSS所做的限制。
7. Margin不一致的问题：
8. 当有多张图片需要排在一行时，通常使用“Float：Left”来实现，这样一来，浏览器就存在兼容性问题。导致图片与后面的内容存在margin不一致的问题。对此一种解决方法就是给图片添加“Display：inline”项即可。
9. DIV居中问题：
10. 通常会利用“vertical-align:middle”来实现，这对于搜狗浏览器来说，是正常的，但是对于IE浏览器来说，却并没有效果。对此，一种较好的解决方法是：将文字的行高设置与DIV一样时即可解决问题。
11. 内外边框合并问题。通常情况下，对于两个相关DIV块，相邻时采用外边距合并原则，其结果只最两个DIV块中Margin最大值做为两个DIV之间的间距。包含的两个DIV之间的间距也遵行同样的规则。
12. 掌握了这一规则，在利用DIV块进行布局时就可以做的更加得心应手。

64、Doctype作用？标准模式与兼容模式有什么区别；

<!DOCTYPE>声明位于位于HTML文档中的第一行，处于 <html> 标签之前。告知浏览器的解析器用什么文档标准解析这个文档。DOCTYPE不存在或格式不正确会导致文档以兼容模式呈现。  
   
标准模式的排版 和JS运作模式都是以该浏览器支持的最高标准运行。在兼容模式中，页面以宽松的向后兼容的方式显示,模拟老式浏览器的行为以防止站点无法工作

65、html5有哪些新特性，移除了哪些元素；

上边有☝☝☝☝☝☝☝☝☝

66、介绍一下标准css盒子模型；

上边有☝☝☝☝☝☝☝☝☝

67、什么事css预处理器；

Less and sass

68、css3有哪些新特性；

1 选择器  
2 RGBA和透明度  
3 多栏布局  
4 多背景图  
5 Word Wrap  
 6 文字阴影  
 7 @font-face属性  
 8 圆角(边框半径)  
  9 边框图片  
  10 盒阴影  
11 盒子大小  
12 媒体查询  
13 语音

69、document.write、innerHTML和innerText 的区别？

document.write是直接写入到页面的内容流，如果在写之前没有调用document.open, 浏览器会自动调用open。每次写完关闭之后重新调用该函数，会导致页面被重写。 innerHTML则是DOM页面元素的一个属性，代表该元素的html内容。你可以精确到某一个具体的元素来进行更改。如果想修改document的内容，则需要修改document.documentElement.innerElement。 innerHTML将内容写入某个DOM节点，不会导致页面全部重绘 innerHTML很多情况下都优于document.write，其原因在于其允许更精确的控制要刷新页面的那一个部分

70、请说出至少三种减少页面加载时间的方法。（加载时间是指感知的时间或者实际加载 的时间）

1、减少http请求（合并文件、合并图片）  
2、使用雪碧图  
3、优化图片文件，减小其尺寸，特别是缩略图，一定要按尺寸生成缩略图然后调用，不要在网页中用resize方法实现，虽然这样看到的图片外形小了，但是其加载的数据量一点也没减少。  
4、 压缩JavaScript、CSS代码  
5、 服务器启用gzip压缩功能  
6、标明高度和宽度  
7、网址后面加上“/”:对服务器而言，不加斜杠服务器会多一次判断的过程，加斜杠就会直接返回网站设置的存放在网站根目录下的默认页面。

71、请描述你曾使用过的css预处理器的优缺点

sass庞大功能全面 所以比较推荐sass less体积小 不推荐使用

72、实现 class为test的div 在屏幕宽为400一下的宽度为200，屏幕宽为400~800的情况下宽 度为350；

1、减少http请求（合并文件、合并图片）  
2、使用雪碧图  
3、优化图片文件，减小其尺寸，特别是缩略图，一定要按尺寸生成缩略图然后调用，不要在网页中用resize方法实现，虽然这样看到的图片外形小了，但是其加载的数据量一点也没减少。  
4、 压缩JavaScript、CSS代码  
5、 服务器启用gzip压缩功能  
6、标明高度和宽度  
7、网址后面加上“/”:对服务器而言，不加斜杠服务器会多一次判断的过程，加斜杠就会直接返回网站设置的存放在网站根目录下的默认页面。  
  
  
 .test{  
          background-color: aqua;  
            width: 350px;  
            height: 35px;  
        }  
        @media screen and (max-width: 400px){  
            .test{  
                    width: 200px;  
            }  
        }

73、实现当屏幕宽度大雨700,小于800是时引用外部样式style-7-9.css

<**link** rel=**"stylesheet"** href=**"style7-7.css"** media=**"screen and (min-width:700px)   
 and (max-width:800px)"**/>

74、HTML中input的6个新属性

button 定义可点击按钮 checkbox 定义复选框

file 定义输入字段和 "浏览"按钮，供文件上传 password 定义密码字段

radio 定义单选按钮 submit 定义提交按钮

75、列一个手机端可以实现手势滑动的插件或类库

Swipe js

76、对WEB标准以及W3C的理解与认识  
  
标签闭合、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；

77、xhtml和html有什么区别

* 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。

78、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?

触发严格模式:  
1.<!-- HTML 4.01 严格型 -->    
2.<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"  "C:\Users\ADMINI~1\AppData\Local\Temp\[5UQ[BL(6~BS2JV6W}N6[%S.pnghttp://www.w3.org/TR/html4/strict.dtd">     
3.    
4.<!-- XHTML 1.0 严格型 -->    
5.<!DOCTYPE html PUBLIC  "-//W3C//DTD XHTML 1.0 Strict//EN"  "C:\Users\ADMINI~1\AppData\Local\Temp\[5UQ[BL(6~BS2JV6W}N6[%S.pnghttp://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">    
触发混杂模式:  
1.<!-- HTML 4.01 过渡型 -->    
2.<!DOCTYPE HTML PUBLIC  "-//W3C//DTD HTML 4.01 Transitional//EN"  "C:\Users\ADMINI~1\AppData\Local\Temp\[5UQ[BL(6~BS2JV6W}N6[%S.pnghttp://www.w3.org/TR/html4/loose.dtd">     
3.    
4.<!-- HTML 4.01 框架集型 -->    
5.<!DOCTYPE HTML PUBLIC  "-//W3C//DTD HTML 4.01 Frameset//EN"  "C:\Users\ADMINI~1\AppData\Local\Temp\[5UQ[BL(6~BS2JV6W}N6[%S.pnghttp://www.w3.org/TR/html4/frameset.dtd">     
6.    
7.<!-- XHTML 1.0 过渡型 -->    
8.<!DOCTYPE html PUBLIC  "-//W3C//DTD XHTML 1.0 Transitional//EN"  "C:\Users\ADMINI~1\AppData\Local\Temp\[5UQ[BL(6~BS2JV6W}N6[%S.pnghttp://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">     
9.    
10.<!-- XHTML 1.0 框架集型 -->    
11.<!DOCTYPE html PUBLIC  "-//W3C//DTD XHTML 1.0 Frameset//EN"  "C:\Users\ADMINI~1\AppData\Local\Temp\[5UQ[BL(6~BS2JV6W}N6[%S.pnghttp://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">    
对于这两种模式之间的差异，最显著的一个例子与Windows上IE专有的盒模型有关。在IE 6出现时，在标准模式中使用的是正确的盒模型，在混杂模式中使用的则是老式的专有盒模型。为了维持对IE 5和更低版本的向后兼容性，Opera 7和更高版本也在混杂模式中使用有缺点的IE盒模型。  
呈现方面的其他差异比较小，而且是与特定浏览器相关的，包括对于十六进制颜色值不需要#号、假设CSS中没有指定单位的长度的单位是像素，以及在使用关键字时将字号增加一级。

79、行内元素有哪些?块级元素有哪些?CSS的盒模型?

行内 a span I b em 块级 p div ul ol dl 标准盒模型 伸缩盒模型（新，旧） 怪异盒模型

80、CSS引入的方式有哪些? link和@import的区别是?

上边有

81、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?

82、标签选择符 类选择符 id选择符

83、标签上title与alt属性的区别是什么?

Title是鼠标滑上显示出来的文字

Alt 是图片资源未正常加载显示出来的文字

84、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.

85、解释css sprites，如何使用。

通过背景定位来实现 也就是background – position，通过定位每个图标的位置而将图片显示出来；

86、浏览器标准模式和怪异模式之间的区别是什么?

所谓的标准模式是指，浏览器按W3C标准解析执行代码；怪异模式则是使用浏览器自己的方式解析执行代码，因为不同浏览器解析执行的方式不一样，所以我们称之为怪异模式。浏览器解析时到底使用标准模式还是怪异模式，与你网页中的DTD声明直接相关，DTD声明定义了标准文档的类型（标准模式解析）文档类型，会使浏览器使用相应的方式加载网页并显示，忽略DTD声明,将使网页进入怪异模式(quirks mode)

87、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：

1、利用浏览器缓存你的 js 和 CSS 文件2、把你的 .js 库文件地址替换成 Google CDN的地址：3、精简和优化你的 js 和 CSS：　4、GZIP 压缩你的 JS 和 CSS 文件：　5、使用css sprites合并图片6、优化你网站图片：

88、清除浮动的几种方式，各自的优缺点

1.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   
2.结尾处加空div标签 clear:both   
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，让人感觉很不好   
3.父级div定义 伪类:after 和 zoom  
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。     
4.父级div定义 overflow:hidden  
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。   
5.父级div定义 overflow:auto   
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。

89、CSS3有哪些新特性？

边框：(Borders) border-color:控制边框颜色，并且有了更大的灵活性，可以产生渐变效果 border-image:控制边框图象 border-corner-image:控制边框边角的图象 border-radius:能产生类似圆角矩形的效果 背景(Backgrounds) background-origin:决定了背景在盒text-shadow:文字投影，可能是因为MAC OSX的Safari浏览器开始支持投影才特意增加的。   
text -overflow:当文字溢出时，用"..."提示。包括ellipsis、clip、ellipsis-word、inherit，前两个CSS2就有了，目前还是部分支持，但有趣的是IE6居然也支持。ellipsis-word可以省略掉最后一个单词，对中文意义不大，inherit可以继承父级元素。 HSL colors:除了支持RGB眼色外，还支持HSL（色相、饱和度、亮度）。以前一般都是作图的时候用HSL色谱，但这样一来会包括更多的颜色。H用度表示，S和L用百分比表示，比如hsl(0,100%, 50%)  
HSLA colors:加了个不透明度(Apacity)，用0到1之间的数来表示，比如hsla(0,100%,50%,0.2)  
opacity:直接控制不透明度，用0到1之间的数来表示  
RGBA colors:和HSLA colors类似，加了个不透明度 Attribute selectors:在属性中可以加入通配符，包括^,$,\*   
[att^=val]:表示开始字符是val的att属性   
[att$=val]:表示结束字符是val的att属性   
[att\*=val]:表示包含至少有一个val的att属性

90、css中的class和id有什么区别。

Class名可以有多个可以重复，id是唯一的

91、请说一下移动端常见的适配不同屏幕大小的方法。

媒体查询 @media screen and（min-width：xxpx） and （max-width:xxpx）{

}

Bootstrap 栅格系统

92、一个高宽未知的图片如何在一个比他大的容器内水平居中。

1.子元素  
  div{text-align:center}  
  p{display:inline-block}  
2.淘宝分页：  
  div{position:relative; left:50%; float:left;}  
  p{position:relative; left:-50%;}  
3.利用diplay:table来解决：  
<div id="box">   
     <span><div id="inner">11111111111111111111</div> </span>  
</div>  
#box{width:100%;height:100%; display:table;text-align:center;}  
#box span{width:100%;display:table-cell; vertical-align:middle;}  
#inner{display:inline-block;width:200px; border:1px solid #000;}  
4.高端的CSS3 flex-box 居中方案  
<div class="parent">  
    <div class="test">asdasdfasdfasdfasdfasdfasdfasdffsd</div>  
</div>  
.parent{position: fixed; top: 0; left: 0; right: 0; bottom: 0; display: -webkit-box;  -webkit-box-align: center;-webkit-box-pack: center;}  
  
.test{-webkit-box-flex: 0;background: #CCC;}

93、Table标签的作用是什么。

表格….

94、定义链接四种状态的伪类的正确书写顺序是？

a:link,a:visited,a:hover,a:active{;}

95、你知道的css选择器有哪些。

元素选择器 id选择器 类选择器 通配符·1

96、遇到疑难问题时，你通常时如何解决的？

百度查询，在解决不了的问公司老员工

97、Inline，inline-block和block的区别是。

Illine：行内元素 使元素可以一行内存在多个，不可设置宽高样式；

inline-block：行内块级元素 一行可以存在多个，可以设置宽高样式；

block：一行只能中存在一个，可以设置样式；

98、页面导入样式时，使用link和@import有什么区别。

上边有

99、CSS超过宽度的文字显示点点，必须要设置的属性。

**overflow**: **hidden**;  
**text-overflow**: **ellipsis**;  
**white-space**: **nowrap**;

100、设置文字阴影属性。设置表格边框合并属性。

Text-shadow,表格边框合并属性border-collapse

101、文本强制换行。

102、display：none与visibility:hidden的区别是什么？

none表示完全没有, 当前的位置会被后面的元素补上来  
hidden仅仅是隐藏, 但是他的位置会保留

103、超链接访问过后hover样式后会出现什么问题？如何解决。

字体会变颜色

104、前端页面有那三部分构成，其作用是什么。

上边有

105、写出至少三个css3中新添加的样式属性的名字及参数。

边框：(Borders) border-color:控制边框颜色，并且有了更大的灵活性，可以产生渐变效果 border-image:控制边框图象 border-corner-image:控制边框边角的图象 border-radius:能产生类似圆角矩形的效果 背景(Backgrounds) background-origin:决定了背景在盒text-shadow:文字投影，可能是因为MAC OSX的Safari浏览器开始支持投影才特意增加的。   
text -overflow:当文字溢出时，用"..."提示。包括ellipsis、clip、ellipsis-word、inherit，前两个CSS2就有了，目前还是部分支持，但有趣的是IE6居然也支持。ellipsis-word可以省略掉最后一个单词，对中文意义不大，inherit可以继承父级元素。 HSL colors:除了支持RGB眼色外，还支持HSL（色相、饱和度、亮度）。以前一般都是作图的时候用HSL色谱，但这样一来会包括更多的颜色。H用度表示，S和L用百分比表示，比如hsl(0,100%, 50%)  
HSLA colors:加了个不透明度(Apacity)，用0到1之间的数来表示，比如hsla(0,100%,50%,0.2)  
opacity:直接控制不透明度，用0到1之间的数来表示  
RGBA colors:和HSLA colors类似，加了个不透明度 Attribute selectors:在属性中可以加入通配符，包括^,$,\*   
[att^=val]:表示开始字符是val的att属性   
[att$=val]:表示结束字符是val的att属性   
[att\*=val]:表示包含至少有一个val的att属性

106、给一个元素添加css样式有哪几种方法说说他们的优先级。

上边有

107、多人项目中，你如何规划css文件样式命名。

108、为了让网页更快的加载，你会如何处理你的css文件和js文件，以及图片文件，页面性 能优化你还知道哪些方法。

上边有

109、你经常遇到的浏览器兼容性有哪些？通常是怎么处理的。

上边有

110、HTML中div与span区别；

在标准文档流里面，块级元素具有以下特点：  
①总是在新行上开始，占据一整行；  
②高度，行高以及外边距和内边距都可控制；  
③宽带始终是与浏览器宽度一样，与内容无关；  
④它可以容纳内联元素和其他块元素。  
行内元素的特点：  
①和其他元素都在一行上；  
②高，行高及外边距和内边距部分可改变；  
③宽度只与内容有关；  
④行内元素只能容纳文本或者其他行内元素。  
不可以设置宽高，其宽度随着内容增加，高度随字体大小而改变，内联元素可以设置外边界，但是外边界不对上下起作用，只能对左右起作用，也可以设置内边界，但是内边界在ie6中不对上下起作用，只能对左右起作用

111、HTML5 存储类型有哪些，以及与他们的区别；

永久性的本地存储（localStorage） 不兼容低端浏览器，需手动清除

会话级别的本地存储（sessionStorage）不兼容低端浏览器，退出页面以后自动清除

Cookie：内存较小

112、css实现垂直水平居中。

1.用inline-block和vertical-align来实现居中  
2.用相对绝对定位和负边距实现上下左右居中：高度和宽度已知  
3.利用绝对定位来实现居中：适合高度，宽度已知的情况。  
4.使用table-cell，inline-block实现水平垂直居中：适合高度宽度未知的情况  
5.使用css3中的transform来时实现水平垂直居中：适合高度宽度未知的情况  
6.还可以使用Flexbox来实现水平垂直居中；适合宽度高度未知情况，但是要注意兼容性

113、浏览器内核分别是什么；

谷歌 Webkit 火狐Mozilla ie trident Opera  Presto Safari webkit

114、列举5种以上表单元素中input的type类型；

1.可以是text。也就是我们常说的文本框，一般用来输入文本，像用户，邮箱。文章标题之类的  
2.可以是password也就是我们常见的密码输入框。password类型的input里面的内容都是\*号或者黑点。任何人是不可见的，只是输入者知道是什么内容。  
  
3.可以是submit，也就是我们常见的提交按钮。点击后，直接触发指定的链接  
4.可以是button，也就是常规的按钮。这个按钮仅是个按钮，没有其它特殊用途。除非我们给他绑定的JS的onclick事件，同样可以做很多事。  
5.可以是reset，也就是重置按钮，我们经常看到很多表单中，有个重新输入。这个就是将FORM内所有表单元素重置到初始值  
6.可以是cancel，也就是取消按钮，用于取消submit的提交  
7.可以是radio，也就是单选按钮。一组按钮中，只能选中一个  
8.可以是checkbox，复选按钮，可以选择多个

115、alt和title分别表示什么含义以及具体应用体现；

图片中

Title是鼠标滑上显示出来的文字

Alt 是图片资源未正常加载显示出来的文字

116、css中position的属性值都有哪些？并描述其含义及具体解释；

position的属性值absolute生成绝对定位的元素，相对于 static 定位以外的第一个父元素进行定位。元素的位置通过 "left", "top", "right" 以及 "bottom" 属性进行规定。  
Relative生成相对定位的元素，相对于其正常位置进行定位。因此，"left:20" 会向元素的 LEFT 位置添加 20 像素。  
fixed生成绝对定位的元素，相对于浏览器窗口进行定位。元素的位置通过 "left", "top", "right" 以及 "bottom" 属性进行规定。  
  
static默认值。没有定位，元素出现在正常的流中（忽略 top, bottom, left, right 或者 z-index 声明）。

117、对html语义化理解，以及语义化有哪些优点。

语义化的HTML就是写出的[HTML代码](https://www.baidu.com/s?wd=HTML%E4%BB%A3%E7%A0%81&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1dWn1n3rym1nyNBrycvryf0IAYqnWm3PW64rj0d0AP8IA3qPjfsn1bkrjKxmLKz0ZNzUjdCIZwsrBtEXh9GuA7EQhF9pywdQhPEUiqkIyN1IA-EUBtkPWbkn1D3n1fsrjmYnHnYrH63" \t "_blank)，符合内容的结构化（内容语义化），选择合适的标签（代码语义化），能够便于开发者阅读和写出更优雅的代码的同时让浏览器的爬虫和机器很好地解析。

优点：

　　1.语义化有利于SEO，有利于搜索引擎爬虫更好的理解我们的网页，从而获取更多的有效信息，提升网页的权重。  
　　2.在没有CSS的时候能够清晰的看出网页的结构，增强可读性。  
　　3.便于团队开发和维护，语义化的HTML可以让开发者更容易的看明白，从而提高团队的效率和协调能力。  
　　4.支持多终端设备的浏览器渲染。

118、经常使用的页面开发工具级测试工具。

1.firefox插件Firebug  
2.SuperPreview微软的网页开发调试利器

119、经常使用什么脚本库，开发或使用什么应用或组件；

用JavaScript脚本开发的轻量级UI组件；Jquery

120、使用css如何让一个宽度为200px的div水平居中，（要求兼容ie，可提供多种方法）；

<!DOCTYPE html>  
<html>  
  
    <head>  
        <meta charset="utf-8" />  
        <title>css</title>  
        <style type="text/css">  
            \* { margin:0; padding:0; list-style:none;  }  
            #vertical\_box {  
                width:200px;  
                display:table;  
                border:1px red solid;  
                height:400px;  
                \*position:relative; /\*针对IE的hack\*/  
            }  
            #vertical\_box\_sub {  
                display: table-cell;  
                vertical-align: middle;  
                \*position:absolute; /\*针对IE的hack\*/  
                \*top:50%;  
            }  
            #vertical\_box\_container {  
                font-family:"宋体";  
                border:1px green solid;  
                \*position:relative; /\*针对IE的hack\*/  
                \*top:-50%;  
                margin:0 auto;  
                width:200px;  
            }  
        </style>  
    </head>  
  
    <body>  
        <div id="vertical\_box">  
            <div id="vertical\_box\_sub">  
                <div id="vertical\_box\_container">  
                    <p>我是居中的文字</p>  
                    <p>我居中</p>  
                    <p>你也居中</p>  
                    <img src="" border=0 alt="\" title="">  
                </div>  
            </div>  
        </div>  
    </body>  
<html>

121、简要画出盒模型，并描述关键要素的含义；

122、页面构造中你遇到过什么样的兼容问题，如何解决

1、针对I E 系列浏览器的网站设计代码        针对 IE 6 的专属 CSS Hack 网站设计代码        #id{        \_display: block;        }            也就是在网站设计CSS属性前加一个小划线就好。        针对 IE 7 的专属 CSS Hack 网站设计代码        #id{        \*display: block;         }        即在网站设计CSS属性前加上一个星号即可        针对 IE 8 的专属 CSS Hack 网站设计代码        #id{        margin-top: 10px 9; /\*IE8\*/        }        如上所示，解决办法为在网站设计CSS属性后分号前加上空格与斜线并加入一个数字9即可  。  
　　2、针对火狐的CSS Hack 网站设计代码        火狐可谓是最标准的浏览器之一了，网站设计技术只要稍稍不到位就能体现得淋漓尽致，所以不少网站设计师也很头痛，其实想要解决火狐的兼容性除了要把网站设计的基础知识扎牢之外只要将CSS代码写入到下图 @-moz-document url-prefix（）{ } 里面就行了        @-moz-document url-prefix()        {         #id{ display: block; }         }    。  
　　3、针对 Safari 的CSS Hack 网站设计代码        Safari是苹果计算机的最新作业系统Mac OS X中新的浏览器，用来取代之前的Internet Explorer for Mac，使用了KDE的KHTML作为浏览器的运算核心。        @media screen and (-webkit-min-device-pixel-ratio:0)        {         #id { display: block; }         }       兼容性做法和火狐相近。  
4、针对 Opera 的CSS Hack 网站设计代码        Opera即Opera Software ASA，是台式机、各种设备和移动网络浏览器市场的商业领袖，因快速、小巧和比其他浏览器更佳的标准兼容性获得了国际上的最终用户和业界媒体的承认，并在网上受到很多人的推崇。        @media all and (-webkit-min-device-pixel-ratio:10000), not all and (-webkit-min-device-pixel-ratio:0)        {        head~body #id { display: block; }        }

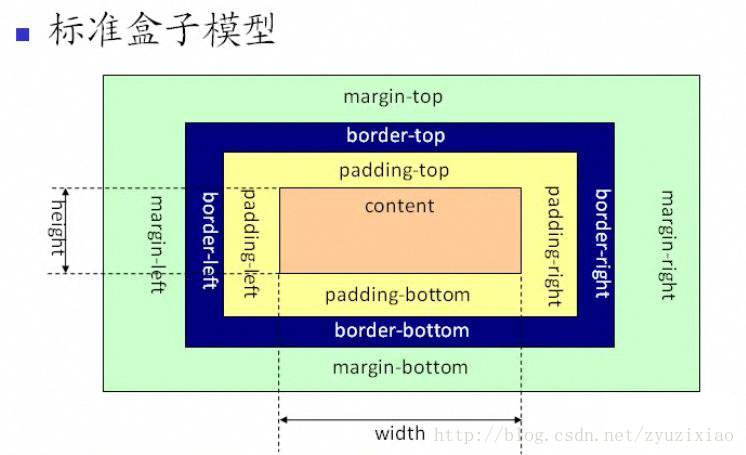
123、Doctype作用？标准模式与兼容模式有什么区别；

上边有！！！

124、html5有哪些新特性，移除了哪些元素；

内容元素，article、footer、header、nav、section。  
表单控件，calendar、date、time、email、url、search。  
控件元素，webworker, websockt, Geolocation。  
移出的元素有下列这些：  
显现层元素：basefont，big，center，font, s，strike，tt，u。  
性能较差元素：frame，frameset，noframes

125、介绍一下标准css盒子模型；



**W3C 盒子模型的范围包括 margin、border、padding、content，并且 content 部分不包含其他部分。**

126、什么事css预处理器；

Css预处理器定义了一种新的语言将Css作为目标生成文件，然后开发者就只要使用这种语言进行编码工作了。预处理器通常可以实现浏览器兼容，变量，结构体等功能，代码更加简洁易于维护。

127、css3有哪些新特性；

上边有！！

128、html5 中的应用缓存是什么？

HTML5 引入了应用程序缓存，这意味着 web 应用可进行缓存，并可在没有因特网连接时进行访问。  
应用程序缓存为应用带来三个优势：  
·离线浏览 - 用户可在应用离线时使用它们  
·速度 - 已缓存资源加载得更快  
·减少服务器负载 - 浏览器将只从服务器下载更新过或更改过的资源。

129、本地存储和cookies'之间的区别是什么

HTML5本地存储是一种让网页可以把键值对存储在用户浏览器客户端的方法。像Cookie一样，这些数据不会因为你打开新网站，刷新页面，乃至关闭你的浏览器而消失。 而与Cookie不同的时，这些数据不会每次随着[HTTP请求](https://www.baidu.com/s?wd=HTTP%E8%AF%B7%E6%B1%82&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YkuW6knWc1njfvPW-bmWn40ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPHR1n1TLrHDknHRvP16YrjDz" \t "_blank)被发送到[服务器端](https://www.baidu.com/s?wd=%E6%9C%8D%E5%8A%A1%E5%99%A8%E7%AB%AF&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YkuW6knWc1njfvPW-bmWn40ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPHR1n1TLrHDknHRvP16YrjDz" \t "_blank)(当然如果你需要这么做，你可以自己编程实现 ）. 因为这是HTML5规范的一部分，这一接口会被浏览器原生支持，不用依赖任何第三方插件。

130、简单介绍boostrap栅格系统

bootstrap3.x使用了四种栅格选项来形成栅格系统，四种栅格选项之间的区别，其实区别只有一条就是适合不同尺寸的屏幕设备。 我们看class前缀这一项，我们姑且以前缀命名这四种栅格选项，他们分别是col-xs、col-sm、col-md、col-lg，我们懂英文的就知 道，lg是large的缩写，md是mid的缩写，sm是small的缩写，xs是\*\*\*的缩写。这样命名就体现了这几种class适应的屏幕宽度不同

131、你如何对网站的文件和资源进行优化？

上边有！！！

132、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  
第一种方法：  
还好，微软提供了这样一个代码：  
<meta http-equiv="x-ua-compatible" content="ie=7" />  
把这段代码放到<head>里面，在ie8里面的页面解析起来就跟ie7一模一样的了，所以，基本上可以无视ie8，剩下的代码只需要这样写就可以了  
background:#ffc; /\* 对firefox有效\*/  
\*background:#ccc; /\* 对ie7有效 \*/  
\_background:#000; /\* 只对ie6有效 \*/  
解释一下吧：  
firefox能解析第一段，后面的两个因为前面加了特殊符号“\*”和“\_”,firefox认不了，所以只认background:#ffc，看到的是黄色；  
ie7前两短都能认，以最后的为准，所以最后解析是background:#ccc，看到的是灰色；  
ie6三段都能认，而且“\_”这个只有ie6能认，所以最后解析是\_background:#000，看到的是黑色  
阿门！已经是最简单和最好理解的写法了，如果你是google进来的，我可以很负责任的告诉你，这种方法是ok的，我测试过。  
ie8的那段兼容7的代码我也测试过了，在我现在的windos 7 测试版所带的ie8是没问题的，以后ie8正式版出来还管不管用就不知道了。  
ps：如果你发现按我这样写还是有问题的话，请查看一下你的html头，看看<head>之前的内容是不是这样的标准写法  
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "C:\Users\ADMINI~1\AppData\Local\Temp\[5UQ[BL(6~BS2JV6W}N6[%S.pnghttp://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">  
<html xmlns="C:\Users\ADMINI~1\AppData\Local\Temp\[5UQ[BL(6~BS2JV6W}N6[%S.pnghttp://www.w3.org/1999/xhtml">  
这个是现在比较规范的写法，如果你是用dreamweaver做页面的话，默认也是这种规范的，切记，非这种规范写法的，兼容性不能保证  
第二种方法：  
要求苛刻的朋友是不愿意在页面头部增加<meta http-equiv="x-ua-compatible" content="ie=7" />这样一句代码的，因为这样的结果是每个页面都得加。那么要想兼容这几个浏览器还真得想别的办法了。早些天本站发布了一篇《完美兼容ie6,ie7,ie8以及firefox的css透明滤镜》，可能当时测试的疏忽，IE8的兼容性没有解决好，好多朋友回复说用不了。今天抽出些时间，查阅大量资料，终于解决了这个问题了。  
以下是兼容IE6/IE7/IE8/FF的写法，注意下面的顺序不可颠倒  
margin-bottom:40px; /\*ff的属性\*/  
margin-bottom:140px\9; /\* IE6/7/8的属性 \*/  
color:red\0; /\* IE8支持 \*/  
\*margin-bottom:450px; /\*IE6/7的属性\*/

133、get和post区别？

1. get是从服务器上获取数据，post是向服务器传送数据。  
2. get是把参数数据队列加到提交表单的ACTION属性所指的URL中，值和表单内各个字段一一对应，在URL中可以看到。post是通过HTTP post机制，将表单内各个字段与其内容放置在HTML HEADER内一起传送到ACTION属性所指的URL地址。用户看不到这个过程。  
3. 对于get方式，[服务器端](https://www.baidu.com/s?wd=%E6%9C%8D%E5%8A%A1%E5%99%A8%E7%AB%AF&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YkPWRYuhP-mHfsnAPhrjfY0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3ErjTdn1RLrH6" \t "_blank)用Request.QueryString获取变量的值，对于post方式，[服务器端](https://www.baidu.com/s?wd=%E6%9C%8D%E5%8A%A1%E5%99%A8%E7%AB%AF&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YkPWRYuhP-mHfsnAPhrjfY0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3ErjTdn1RLrH6" \t "_blank)用Request.Form获取提交的数据。  
4. get传送的数据量较小，不能大于2KB。post传送的数据量较大，一般被默认为不受限制。但理论上，IIS4中最大量为80KB，IIS5中为100KB。  
5. get安全性非常低，post安全性较高。但是执行效率却比Post方法好。

134、Doctype的作用是什么？

上边有

135、浏览器标准模式和怪异模式的区别是什么。

标准模式的排版 和JS运作模式都是以该浏览器支持的最高标准运行。在兼容模式中，页面以宽松的向后兼容的方式显示,模拟老式浏览器的行为以防止站点无法工作。

136、解释一下浮动和它的工作原理。

父元素的高度无法被撑开，影响与父元素同级的元素  
与浮动元素同级的非浮动元素（内联元素）会跟随其后  
若非第一个元素浮动，则该元素之前的元素也需要浮动，否则会影响页面显示的结构

137、列举不同的清除浮动的方法，并指出他们各自适用的场景。

使用空标签清除浮动

使用overflow属性

使用after伪对象清除浮动

**第二阶段**

1. jQuery this与this区别
2. input name a与p name a 这两个如何用jQuery定位到input
3. $("input[name='a']")

3、请描述一下 cookies，sessionStorage 和 localStorage 的区别？

永久性的本地存储（localStorage） 不兼容低端浏览器，需手动清除

会话级别的本地存储（sessionStorage）不兼容低端浏览器，退出页面以后自动清除

Cookie：内存较小

1. 对BFC规范的理解？

BFC是Block Formatting Context （块级格式化上下文）的缩写，它是W3C CSS 2.1 规范中的一个概念，是一个独立的渲染区域，只有Block-level box参与， 它规定了内部的Block-level Box如何布局，并且与这个区域外部毫不相干。

5、JavaScript原型，原型链 ? 有什么特点？

6、说几条写JavaScript的基本规范？

建议使用\*\*四个空格\*\*缩进，这也是JSLint中默认的缩进。

## 花括号{} ##

\*\*建议：总是使用花括号，并始终把在与之前的语句放在同一行：\*\*

最后需要注意的一个空格：左花括号前最好使用空格！

7、null，undefined 的区别？

在JavaScript中，null是字面量同时也是语言中的关键字，用来表示无法识别的对象值。换句话说，这用来表示“无值（no value）”。虽然相似，undefined实际上代表了不存在的值（non-existence of a value）。都是完全不可变的，没有属性和方法，也不能给其属性赋值。事实上,试图访问或定义一个属性将会引发一个类型错误（TypeError）。正如他们的名字暗示的那样，他们是完全无效的值。

8、谈谈This对象的理解。

1：this谁调用指向谁

2.闭包中的this指向window

3.setInterval和setTimeout中的this指向window

4.事件中的this

HTML：this指向window

DOMO：this指向绑定该事件的元素

DOM2级：IE中指向window，非IE中指向绑定该事件的元素

5.自执行函数this指向window

6.可以借助call和apply来改变this指向

9、事件是？IE与火狐的事件机制有什么区别？ 如何阻止冒泡？

10、什么是闭包（closure），为什么要用它？

闭包是可以包含自由（未绑定到特定对象）变量的代码块；这些变量不是在这个代码块内或者任何全局上下文中定义的，而是在定义代码块的环境中定义（局部变量）。“闭包” 一词来源于以下两者的结合：要执行的代码块（由于自由变量被包含在代码块中，这些自由变量以及它们引用的对象没有被释放）和为自由变量提供绑定的计算环境（作用域）。在 Scala、Scheme、Common Lisp、Smalltalk、Groovy、JavaScript、Ruby、 Python、Lua、objective c 以及Java（Java8及以上）等语言中都能找到对闭包不同程度的支持。 中文名：闭包 外文名：closure 相关学科：[离散数学](https://www.baidu.com/s?wd=%E7%A6%BB%E6%95%A3%E6%95%B0%E5%AD%A6&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Ykm1fYnj-9nHFhrHRzmH9h0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnHn3PjnknH63PHb4n1ndrH0vn0" \t "_blank) 用途：编程逻辑 特点：未绑定到特定对象 分享 拓扑概念 集合A的闭包定义为所有包含A的闭集之交。A的闭包是包含A的最小闭集。 本质 集合 S 是闭集[当且仅当](https://www.baidu.com/s?wd=%E5%BD%93%E4%B8%94%E4%BB%85%E5%BD%93&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Ykm1fYnj-9nHFhrHRzmH9h0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnHn3PjnknH63PHb4n1ndrH0vn0" \t "_blank) Cl(S)=S（这里的cl即closure，闭包）。特别的，空集的闭包是空集，X 的闭包是 X。集合的交集的闭包总是集合的闭包的交集的子集（不一定是[真子集](https://www.baidu.com/s?wd=%E7%9C%9F%E5%AD%90%E9%9B%86&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Ykm1fYnj-9nHFhrHRzmH9h0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnHn3PjnknH63PHb4n1ndrH0vn0" \t "_blank)）。有限多个集合的并集的闭包和这些集合的闭包的并集相等；零个集合的并集为空集，所以这个命题包含了前面的空集的闭包的特殊情况。无限多个集合的并集的闭包不一定等于这些集合的闭包的并集，但前者一定是后者的父集。 若 A 为包含 S 的 X 的子空间，则 S 在 A 中计算得到的闭包等于 A 和 S 在 X 中计算得到的闭包（Cl\_A(S) = A ∩ Cl\_X(S））的交集。特别的，S在 A 中是稠密的，[当且仅当](https://www.baidu.com/s?wd=%E5%BD%93%E4%B8%94%E4%BB%85%E5%BD%93&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Ykm1fYnj-9nHFhrHRzmH9h0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnHn3PjnknH63PHb4n1ndrH0vn0" \t "_blank) A 是 Cl\_X(S) 的子集。

11、new操作符具体干了什么呢?

new共经过了4几个阶段

1、创建一个空对象

**[javascript]** [view plain](http://blog.csdn.net/lxcao/article/details/52792466" \o "view plain) [copy](http://blog.csdn.net/lxcao/article/details/52792466" \o "copy)

[在CODE上查看代码片](https://code.csdn.net/snippets/1924148)

1. varobj=**new** Object();

2、设置原型链

**[javascript]** [view plain](http://blog.csdn.net/lxcao/article/details/52792466" \o "view plain) [copy](http://blog.csdn.net/lxcao/article/details/52792466" \o "copy)

[在CODE上查看代码片](https://code.csdn.net/snippets/1924148)

1. obj.\_\_proto\_\_= Func.prototype;

3、让Func中的this指向obj，并执行Func的函数体。

**[javascript]** [view plain](http://blog.csdn.net/lxcao/article/details/52792466" \o "view plain) [copy](http://blog.csdn.net/lxcao/article/details/52792466" \o "copy)

[在CODE上查看代码片](https://code.csdn.net/snippets/1924148)

1. **var** result =Func.call(obj);

4、判断Func的返回值类型：

如果是值类型，返回obj。如果是引用类型，就返回这个引用类型的对象。

**[javascript]** [view plain](http://blog.csdn.net/lxcao/article/details/52792466" \o "view plain) [copy](http://blog.csdn.net/lxcao/article/details/52792466" \o "copy)

[在CODE上查看代码片](https://code.csdn.net/snippets/1924148)

1. **if** (**typeof**(result) == "object"){
2. func=result;
3. }
4. **else**{
5. func=obj;;
6. }

12、JavaScript中的作用域与变量声明提升？

作用域：一个变量的作用域表示这个变量存在的上下文。它指定了你可以访问哪些变量以及你是否有权限访问某个变量。  
变量作用域分为局部作用域和全局作用域。  
变量声明提升：所以的变量声明都会提升到函数的开头（如果这个变量在这个函数里面）或者全局作用域的开头（如果这个变量是一个全局变量）。

13、如何编写高性能的Javascript？

**1、尽量不要用for-in 循环去访问数组，建议用 for 循环进行循环**

**2、建议将对象进行缓存处理，特别是DOM访问是比较消耗资源的：**

**3、建议不要在函数内进行过深的嵌套判断：**

**4、避免循环引用，防止内存泄漏：**

**5、建议避免在函数内返回一个未声明的变量，会污染外部变量：**

**6、var声明变量，建议写在多行**

14、那些操作会造成内存泄漏？

1.常发性内存泄漏   发生内存泄露的代码会被多次执行到，每次执行的时候都会导致一块内存泄露  
2.偶发性内存泄露    发生内存泄漏的代码只有在某些特定环境或操作过程下才会发生。常发性和偶发性是相对的。对于特定的环境，偶发性的也许就变成了常发性的。所以  
测试环境和测试方法对检测内存泄漏至关重要。  
3.一次性内存泄漏。　发生内存泄漏的代码只会被执行一次，或者由于算法上的缺陷，导致总会有一块仅且一块内存发生泄漏。比如，在类的构造函数中分配内存，在析构函数中却没有释放该内存，所以内存泄漏只会发生一次。  
4.隐式内存泄漏。　程序在运行过程中不停的分配内存，但是直到结束的时候才释放内存。严格的说这里并没有发生内存泄漏，因为最终程序释放了所有申请的内存。但是对于一个服务器程序，需要运行几天，几周甚至几个月，不及时释放内存也可能导致最终耗尽系统的所有内存。所以，我们称这类内存泄漏为隐式内存泄漏。

15、你遇到过比较难的技术问题是？你是如何解决的？

16、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？

用JavaScript脚本开发的轻量级UI组件；Jquery

17、页面重构怎么操作？

1. 结构完整，可通过标准验证

2. 标签语义化，结构合理

3. 充分考虑到页面在站点中的“作用和重要性”，并对其进行有针对性的优化

18、什么叫优雅降级和渐进增强？

  渐进增强 progressive enhancement：针对低版本浏览器进行构建页面，保证最基本的功能，然后再针对高级浏览器进行效果、交互等改进和追加功能达到更好的用户体验  
      优雅降级 graceful degradation：一开始就构建完整的功能，然后再针对低版本浏览器进行兼容  
区别：优雅降级是从复杂的现状开始，并试图减少用户体验的供给，而渐进增强则是从一个非常基础的，能够起作用的版本开始，并不断扩充，以适应未来环境的需要。降级（功能衰减）意味着往回看；而渐进增强则意味着朝前看，同时保证其根基处于安全地带。

19、WEB应用从服务器主动推送Data到客户端有那些方式？

主要就是post和get两种传递方式

20、编程实现javascript在String中写一个trim，要求能够去除一个字符串开始和结尾的空格

var str = '      dsafasfasf        ';

    var qkg = str.trim();

    console.log(qkg)

21、json的了解

[JSON](http://baike.baidu.com/view/136475.htm" \t "_blank)([JavaScript](http://baike.baidu.com/view/16168.htm" \t "_blank) Object Notation) 是一种轻量级的数据交换格式。它基于[ECMAScript](http://baike.baidu.com/view/810176.htm" \t "_blank)的一个子集。 JSON采用完全独立于语言的文本格式，但是也使用了类似于C语言家族的习惯（包括[C](http://baike.baidu.com/subview/10075/6770152.htm" \t "_blank)、C++、[C#](http://baike.baidu.com/view/6590.htm" \t "_blank)、[Java](http://baike.baidu.com/subview/29/12654100.htm" \t "_blank)、JavaScript、[Perl](http://baike.baidu.com/view/46614.htm" \t "_blank)、[Python](http://baike.baidu.com/view/21087.htm" \t "_blank)等）。这些特性使JSON成为理想的数据交换语言。 易于人阅读和编写，同时也易于机器解析和生成(一般用于提升网络传输速率)。

22、一个页面从输入url 到页面加载显示完成，这个过程中都发生什么了

这个过程可以分为四个步骤：  
1.当发送一个URL请求时，不管这个URL是web页面的URL还是web页面的每个资源的URL，浏览器都会开启一个线程来处理这个请求，同时在远程DNS服务器上启动一个DNS查询，这样就可以使得浏览器获得请求对应的IP地址了。  
2.浏览器通过与远程web服务器TCP三次握手协商来建立一个TCP/IP链接。三个握手包含一个同步报文，一个同步-应答报文和一个应答报文，这三个报文在浏览器和服务器之间进行传递，该握手首先由客户端尝试建立起通信，而后服务器应答并接受客户端的请求，最后由由客户端发出该请求已经被接受的报文。  
3.一旦TCP/IP链接建立，浏览器会通过该链接向远程服务器发送HTTP的GET请求。远程服务器找到资源并使用HTTP响应返回该资源，值为200的HTTP状态码表示一个正确的响应。  
4.此时，web服务器提供资源服务，客户端开始下载资源。

23、javascript如何创建一个对象，？{画出次对象的内存图}

（1）使用内置对象   
（2）使用JSON符号   
（3）自定义对象构造

JavaScript可用的内置对象可分为两种：   
1，JavaScript语言原生对象（语言级对象），如String、Object、Function等；   
2，JavaScript运行期的宿主对象（环境宿主级对象），如window、document、body等。

24、juery中如何将数组转化为json字符串，然后在转化回来

$.fn.stringify = function() {  
  return JSON.stringify(this);  
}  
$(arry).stringify();

25、说一说 最近流行的一些新的东西 》常去哪些网站？

26、给juery扩展一个新的空方法，方法名称为newFunc

<script src="jquery/jQuery.1.7.2.js"></script>

<script>

$(function () {

$.newFunc=function(xin){

alert(xin)

};

$.newFunc(2222)

})

</script>

27、编程实现使用JavaScript实现一个深拷贝方法

28、请写出超链接的顺序或你在初级样式中的连接方法

定义链接样式的四个伪类，它们分别是：  
:link   
:visited   
:hover   
:active   
因为我们要定义链接样式，所以其中必不可少的就是超级链接中的锚标签--a，锚标签和伪类链接起来书写的方法就是定义链接样式的基础方法，它们的写法如下：  
a:link，定义正常链接的样式；   
a:visited，定义已访问过链接的样式；   
a:hover，定义鼠标悬浮在链接上时的样式；   
a:active，定义鼠标点击链接时的样式。

29、函数几种定义方法

JS函数的定义方式比较灵活,它不同于其他的语言,每个函数都是作为一个对象被维护和运行的.先看几种常用的定义方式：function func1([参数]){/\*函数体\*/}var func2=function([参数]){/\*函数体\*/};var func3=function func4([参数]){/\*函数体\*/};var func5=new Function();上述第一种方式是最常用的方式,不用多说.第二种是将一匿名函数赋给一个变量,调用方法：func2([函数]);第三种是将func4赋给变量func3,调用方法：func3([函数]);或func4([函数]);第四种是声明func5为一个对象.再看看它们的区别：function func(){//函数体}//等价于var func=function(){//函数体}但同样是定义函数,在用法上有一定的区别.//这样是正确的func(1);function func(a){alert(a);}//这样是错误的,会提示func未定义,主要是在调用func之前没有定义func(1);var func = function(a){alert(a);}//这样是正确的,在调用func之前有定义var func = function(a){alert(a);}func(1);用同样的方法可以去理解第三种定义方式.

30、JavaScript的typeof 返回哪些数据类型

typeof 返回值有六种可能： "number," "string,",”boolean," "object," "function," 和 "undefined."

31、写一个获取非行间样式的函数

32、时间委托1.减少代码量，提高性能  
2.借助e.target和冒泡机制把事件绑定在上级元素上事件由上级元素监听由子元素触发  
3.可以进行事件的监听，得到事件的真正触发者  
4.后绑定元素时解决事件未绑定的问题

是什么 有什么好处

33、jquery 中哪几种类型选择器？

(1)基本

(2)层次选择器：

(3)基本过滤器选择器

(4)内容过滤器选择器

(5)可见性过滤器选择器

(6)属性过滤器选择器

(7)子元素过滤器选择器

(8)表单选择器

(9)表单过滤器选择器

34、闭包是什么 有什么特性 对页面有什么影响

闭包：函数内部返回一个函数，返回出来的函数被我们称之为闭包  
1可以读取函数内部的变量2让这些变量始终保持在内存中  
会导致页面加载时内存消耗很大，IE会导致内在泄露，因此尽量少用或用时要及时删除变量。

35、如何阻止事件冒泡和默认事件

 代码如下:

function stopBubble(e){

  if(e&&e.stopPropagation){//非IE

   e.stopPropagation();

  }

  else{//IE

   window.event.cancelBubble=true;

  }

 }

<a onclick=''>如果要阻止默认事件的触发，即默认的href事件，那么就需要调用如下函数：

代码如下:

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;

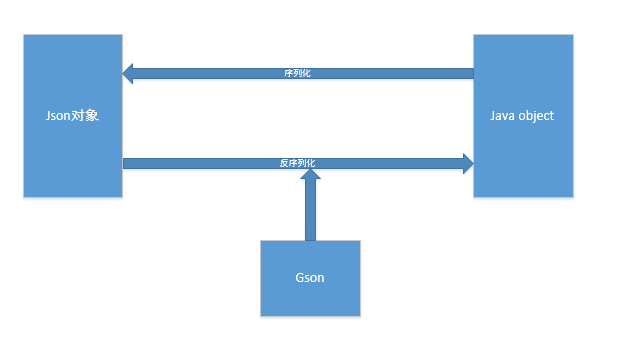
         return false;

     }

36、添加 删除 替换 插入 到某个节点的方法

obj.appendChidl()  
obj.innersetBefore  
obj.replaceChild  
obj.removeChild

37、解释json 的原理 以及为什么不是真正的ajax



38、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  
38.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)

39、js编写一个数组去重的方法

Array.prototype.unique1 = function(){  
 var res = [this[0]];  
 for(var i = 1; i < this.length; i++){  
  var repeat = false;  
  for(var j = 0; j < res.length; j++){  
   if(this[i] == res[j]){  
    repeat = true;  
    break;  
   }  
  }  
  if(!repeat){  
   res.push(this[i]);  
  }  
 }  
 return res;  
}  
var arr = [1, 'a', 'a', 'b', 'd', 'e', 'e', 1, 0]  
alert(arr.unique1());

40、split（）join（）的 区别 数组方法pop（）push（）unshift（）shift（）

Join 函数获取一批字符串，然后用分隔符字符串将它们联接起来，从而返回一个字符串。Split 函数获取一个字符串，然后在分隔符处将其断开，从而返回一批字符串。但是，这两个函数之间的主要区别在于 Join 可以使用任何分隔符字符串将多个字符串连接起来，而 Split 只能使用一个字符分隔符将字符串断开。

41、事件绑定和普通绑定有什么区别

<!doctype html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<title>Document</title>

</head>

<body>

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

<script type="text/javascript">

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('事件绑定1');//弹出

},false);

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

alert('事件绑定2');//弹出

},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。

</script>

</body>

</html>

42、ECMAScript和JavaScript的区别？

ECMA是European Computer Manufacturers Association的缩写，即欧洲计算机制造商协会。欧洲计算机制造商协会是制定信息传输与通讯的国际化标准组织。  
ECMAScript是ECMA制定的标准化脚本语言。  
目前JavaScript使用的ECMAScript版本为ECMAScript-262。  
ECMAScript被国际化标准组织采纳为ISO/IEC 16262。

43、break和continue区别？

break是结束整个循环体，continue是结束单次循环

44、字符串中substring、substr、slice区别？

① 从定义上看： substring和slice是同类的，参数都是字符串的某个｛开始｝位置到某个｛结束｝位置（但｛结束｝位置的字符不包括在结果中）；而substr则是字符串的某个｛开始｝位置起，数length个长度的字符才结束。－－ 共性：从start开始，如果没有第2个参数，都是直到字符串末尾。  
② substring和slice的区别则是，slice可以接受“负数”，表示从字符串尾部开始计数； 而substring则把负数或其它无效的数，当作0。  
③ substr的start也可接受负数，也表示从字符串尾部计数，这点和slice相同；但substr的length则不能小于1，否则返回空字符串。  
如果只知道位置，那么用slice；如果知道长度，则用substr。

45、清除字符串前后空格，要兼容所有浏览器？

去掉空格可以用trim()函数，  
比如   
String aa=" xx ";  
aa=aa.trim();  
去掉字符也可以使用 replace或者replaceAll  
比如  
String aa=" aaxa ";  
aa=aa.replaceAll("x","");//意思就是把x这个字符替换成空，  
当然你这个也可以用来替换空格  
aa=aa.replaceAll(" ","");//意思就是把 这个空字符替换成空，

46、push、pop、shift和unshift 区别？

shift:从集合中把第一个元素删除，并返回这个元素的值。  
unshift: 在集合开头添加一个或更多元素，并返回新的长度  
push:在集合中添加元素，并返回新的长度  
pop:从集合中把最后一个元素删除，并返回这个元素的值。

47、数组和字符串之间的转换方式？

**一、数组转字符串**

需要将数组元素用某个字符连接成字符串，示例代码如下：

[?](http://www.jb51.net/article/52038.htm)

|  |  |
| --- | --- |
| 1  2  3 | var a, b;  a = new Array(0,1,2,3,4);  b = a.join("-"); |

**二、字符串转数组**

实现方法为将字符串按某个字符切割成若干个字符串，并以数组形式返回，示例代码如下：

[?](http://www.jb51.net/article/52038.htm)

|  |  |
| --- | --- |
| 1  2 | var s = "abc,abcd,aaa";  ss = s.split(",");// 在每个逗号(,)处进行分解。 |

48、数组排序？

1、  
 function maopao(arr) {  
        for(var i=0;i<arr.length;i++){            for(var j=0;j<arr.length-1-i;j++){//依次比较响铃2项  
  
                if(arr[j]>arr[j+1]){  
                    var temp = arr[j+1];  
                    arr[j+1] = arr[j];  
                    arr[j] = temp;  
                }  
            }  
  
        }  
        return arr;  
    }  
console.log(maopao(arr));  
  
2、  
var arr = [423,42,3,35,436,45,6,4343,456,65,353,465,242];  
    function  fast(arr) {  
       if(arr.length<=1){  
           return arr;  
       }else {  
           var index = Math.floor(arr.length/2);  
           var zj = arr.splice(index,1)[0];  
           var left = [];  
           var right = [];  
           for(var i=0;i<arr.length;i++){  
               if(arr[i]>zj){  
                   right.push(arr[i]);  
               }else {  
                   left.push(arr[i]);  
               }  
           }  
           return fast(left).concat(zj,fast(right));  
       }  
    }  
    console.log(fast(arr));

49、自执行函数两种写法？

　①第一种方法

　　将如上代码改为：

<SCRIPT LANGUAGE="JavaScript">

　　functionn MyAutoRun()

　　{

　　　//以下是您的函数的代码，请自行修改先！

　　　alert("函数自动执行哦！");

　　}

　　window.onload=MyAutoRun(); //仅需要加这一句

</SCRIPT>

　　②第二种方法

　　修改网页的Body为：

　　<body onLoad="MyAutoRun();">

　　或者改为：

　　<body onLoad="javascript:MyAutoRun();">

　　③第三种方法

　　使用JS定时器来间断性的执行函数：

　　setTimeout("MyAutoRun()",1000); //隔1000毫秒就执行一次MyAutoRun()函数

　　实现方法，将最上面的那JS函数，改为：

<SCRIPT LANGUAGE="JavaScript">

　　functionn MyAutoRun()

　　{

　　　//以下是您的函数的代码，请自行修改先！

　　　alert("函数自动执行哦！");

　　}

　　setTimeout("MyAutoRun()",1000); //这样就行拉

50、取10-100之间的一个随机数？取“a-zA-Z”中的4个随机数？

 var num = Math.floor(Math.random() \* 90) + 10;  
  
    var str = 'qwertyuiopasdfghjklzxcvbnmQWERTYUIOPASDFGHJKLZXCVBNM';  
    var newstr = '';  
    for (var i = 0; i < 4; i++) {  
        var n = Math.floor(Math.random() \* 51);  
        newstr = newstr + str[n];  
    }

51、输出当前日期的前一天？（2016-5-17，则输出2016-5-16）

52、js中选择器的几种方法？

1、通过ID选取元素（getElementById）  
2、通过名称name选取元素（getElementsByName）  
3、通过标签名选取元素（getElementsByTagName）  
4、通过CSS类选取元素（getElementsByClassName）  
5、通过CSS选择器选取元素document.querySelectorAll("selector")/  
document.querySelector

53、创建节点方法？

//$("p").append("<b>你好吗？</b>");//向p元素中追加《b》   
//$("<b>你好吗？</b>").appendTo("p");//将《b》追加到p元素中   
//$("p").prepend("<b>你好吗？</b>");//向p中前置《b》   
//$("<b>你好吗？</b>").prependTo("p");//将《b》前置到p元素中   
//$("p").after("<b>你好吗？</b>");//向p元素后插入《b》   
//$("<b>你好吗？</b>").insertAfter("p");//将《b》插入到p元素后边   
//$("p").before("<b>你好吗？</b>");//在p元素之前添加《b》   
//$("<b>你好吗？</b>").insertBefore("p");//将《b》插入到p元素前面

54、DOM 0级和2级事件的写法、清除方法及优缺点？

DOM 0：一是在标签内写onclick事件二是在JS写onlicke=function（）{}函数  
2级只有一个： 监听方法，有两个方法用来添加和移除事件处理程序：addEventListener()和removeEventListener()。  
它们都有三个参数：第一个参数是事件名（如click）；第二个参数是事件处理程序函数；第三个参数如果是true则表示在捕获阶段调用，为false表示在冒泡阶段调用。  
只有2级DOM包含3个事件：事件捕获阶段、处于目标阶段和事件冒泡阶段  
DOM0级事件  
obj.onclick = function(){};  
取消obj.onclick = null;  
DOM2级事件  
addEventListener(event,fn,onOff);  
取消事件removeEventListener(event,fn,onOff);  
0级的话没有兼容性的问题但是一个元素只能绑定一个元素   
2级的话有兼容性的问题ie和其他浏览器不一样,其他浏览器可以设置事件是否冒泡但ie只能冒泡,支持绑定多个事件,但ie是从后往前执行的

55、target和currentTarget区别？

currentTarget始终是监听事件者，而target是事件的真正发出者

56、数组去重方法？

var arr=[1,1,2,2,'ha','ha'];  
    function qc(arr){  
        if(arr.length<=1){  
            return arr;  
        }else{  
            var obj={};  
            for(var i=0;i<arr.length;i++){  
                obj[arr[i]]=1;  
            }  
        }  
        return Object.keys(obj);  
  
    }  
    console.log(qc(arr));

57、回流和重绘的解释、关系及区别？

1. 当render tree中的一部分(或全部)因为元素的规模尺寸，布局，隐藏等改变而需要重新构建。这就称为回流(reflow)。每个页面至少需要一次回流，就是在页面第一次加载的时候。在回流的时候，浏览器会使渲染树中受到影响的部分失效，并重新构造这部分渲染树，完成回流后，浏览器会重新绘制受影响的部分到屏幕中，该过程成为重绘。

2. 当render tree中的一些元素需要更新属性，而这些属性只是影响元素的外观，风格，而不会影响布局的，比如background-color。则就叫称为重绘。

注意：回流必将引起重绘，而重绘不一定会引起回流

58、怎样能减少页面的回流和重绘？

1. 直接改变className，如果动态改变样式，则使用cssText（考虑没有优化的浏览器）  
2. 让要操作的元素进行”离线处理”，处理完后一起更新  
3.不要经常访问会引起浏览器flush队列的属性，如果你确实要访问，利用缓存  
4. 让元素脱离动画流，减少回流的Render Tree的规模

59、两种定时器的写法及清除？

window.setTimeout([function],[interval]) 设置一个定时器,并且设定了一个等待的时间[interval],当到达时间后,执行对应的方法[function],当方法执行完成定时器停止(但是定时器还在,只不过没用了);  
  
window.setInterval([function],[interval]) 设置一个定时器,并且设定了一个等待的时间[interval],当到达时间后,执行对应的方法[function],当方法执行完成,定时器并没有停止,以后每隔[interval]这么长的时间都会重新的执行对应的方法[function],直到我们手动清除定时器为止;

清除：

window.clearInterval(timer1)/window.clearTimeout(time1)；两种清除方式都可以清除通过setTimeout和setInterval设置的定时器(两种方式在设置定时器的时候有区别，清除定时器的时候没有区别)，并且参数不仅可以是timer，还可以是其返回值，例如1,2；需要注意的是，定时器即使清除了，其返回值也不会清除，之后设置的定时器的返回值也会在其返回值的基础上继续向后排，这点类似于银行的排队领号，即使1号的业务办理完了，后边的人仍是从2号开始继续领号，而不是重新从1开始；

60、判断是手机还是PC？

1、第一种：  
function IsPC() {  
    var userAgentInfo = navigator.userAgent;  
    var Agents = ["Android", "iPhone",  
                "SymbianOS", "Windows Phone",  
                "iPad", "iPod"];  
    var flag = true;  
    for (var v = 0; v < Agents.length; v++) {  
        if (userAgentInfo.indexOf(Agents[v]) > 0) {  
            flag = false;  
            break;  
        }  
    }  
    return flag;  
}  
2、第二种：  
function browserRedirect() {  
    var sUserAgent = navigator.userAgent.toLowerCase();  
    var bIsIpad = sUserAgent.match(/ipad/i) == "ipad";  
    var bIsIphoneOs = sUserAgent.match(/iphone os/i) == "iphone os";  
    var bIsMidp = sUserAgent.match(/midp/i) == "midp";  
    var bIsUc7 = sUserAgent.match(/rv:C:\Users\ADMINI~1\AppData\Local\Temp\`7_{~]GF$3{MOQ4V_}PH]YC.png1.2.3.4/i) == "rv:C:\Users\ADMINI~1\AppData\Local\Temp\%W@GJ$ACOF(TYDYECOKVDYB.png1.2.3.4";  
    var bIsUc = sUserAgent.match(/ucweb/i) == "ucweb";  
    var bIsAndroid = sUserAgent.match(/android/i) == "android";  
    var bIsCE = sUserAgent.match(/windows ce/i) == "windows ce";  
    var bIsWM = sUserAgent.match(/windows mobile/i) == "windows mobile";  
    if (!(bIsIpad || bIsIphoneOs || bIsMidp || bIsUc7 || bIsUc || bIsAndroid || bIsCE || bIsWM) ){  
        window.location.href=B页面;  
    }  
}  
browserRedirect();

61、获取数组中的最大值？

首先把数组的第一个数赋值给变量max和min，分别表示最大值和最小值，再依次判断数组的其他，判断当前的最大值和最小值是否就是最大值和最小值，如果不是进行替换，最后输出最大值和最小值。  
#include<stdio.h>  
#define N 6  
int main()  
{  
 int a[N],max,min,i;  
 for(i=0;i<N;i++)//输入数组   
 scanf("%d",&a[i]);  
 max=min=a[0];//把数组第一个数赋值给max和min   
 for(i=1;i<N;i++){//依次判断剩下每个数   
 if(a[i]>max) max=a[i];//如果该数大于max,把其赋值给max   
 if(a[i]<min) min=a[i];//如果该数小于min,把其赋值给min   
 }  
 printf("最大值：%d，最小值：%d\n",max,min);  
 return 0;  
}

62、写一个改变this指向的闭包？

var obj={

d:5,

e:6

};

var d=6;

function waibu(){

return function(){

var a=5;

var c=a+this.d;

return c;

}

}

var neibu=waibu();

alert(neibu.call(obj))

63、使用正则写一个验证邮箱的表达式？

function CheckMail(mail) {  
 var filter = /^([[a-z](https://www.baidu.com/s?wd=a-z&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YkPvPBP179njcdnj0LrARd0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnHc1rjnvrjfvn1fsPHfLPH6drf" \t "_blank)[A-Z](https://www.baidu.com/s?wd=A-Z&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YkPvPBP179njcdnj0LrARd0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnHc1rjnvrjfvn1fsPHfLPH6drf" \t "_blank)0-9\_\.\-])+\@(([[a-z](https://www.baidu.com/s?wd=a-z&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YkPvPBP179njcdnj0LrARd0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnHc1rjnvrjfvn1fsPHfLPH6drf" \t "_blank)A-Z0-9\-])+\.)+([a-zA-Z0-9]{2,4})+$/;  
 if (filter.test(mail)) return true;  
 else {  
 alert('您的电子邮件格式不正确');  
 return false;}  
}

64、sessionStorage、localStorage、cookie的区别？

sessionStorage 和 localStorage 是HTML5 Web Storage API 提供的，可以方便的在web请求之间保存数据。有了本地数据，就可以避免数据在浏览器和服务器间不必要地来回传递。  
sessionStorage、localStorage、cookie都是在浏览器端存储的数据，其中sessionStorage的概念很特别，引入了一个“浏览器窗口”的概念。sessionStorage是在同源的同窗口（或tab）中，始终存在的数据。也就是说只要这个浏览器窗口没有关闭，即使刷新页面或进入同源另一页面，数据仍然存在。关闭窗口后，sessionStorage即被销毁。同时“独立”打开的不同窗口，即使是同一页面，sessionStorage对象也是不同的。

65、添加、删除、替换、插入到某个节点的方法？

<html>   
<head>   
   <meta http-equiv="Content-Type" content="text/html; charset=utf-8">   
   <title>jQuery插入,复制、替换和删除节点</title>   
<script type="text/javascript" src="jquery-1.3.2.js"></script>   
<script type="text/javascript">   
   $(document).ready(function(){   
     //几种添加节点的方法   
    //$("p").append("<b>你好吗？</b>");//向p元素中追加《b》   
    //$("<b>你好吗？</b>").appendTo("p");//将《b》追加到p元素中   
    //$("p").prepend("<b>你好吗？</b>");//向p中前置《b》   
    //$("<b>你好吗？</b>").prependTo("p");//将《b》前置到p元素中   
    //$("p").after("<b>你好吗？</b>");//向p元素后插入《b》   
    //$("<b>你好吗？</b>").insertAfter("p");//将《b》插入到p元素后边   
    //$("p").before("<b>你好吗？</b>");//在p元素之前添加《b》   
    //$("<b>你好吗？</b>").insertBefore("p");//将《b》插入到p元素前面   
  
     
      //几种删除节点的方法   
    //var $li=$("ul li:eq(1)").remove();//删除ul节点中第2个元素节点   
    //$("ul").append($li);//把刚删除的元素节点从新添加到ul元素中去   
    //$("ul li").remove("li[title!=菠萝]");//将ul元素下title属性不等于"菠萝"的li元素删除   
    //$("ul li:eq(1)").empty();//清空ul节点下第2个li元素的内容   
  
    //复制节点   
   /\* $("ul li").click(function(){   
     $(this).clone(true).appendTo("ul");//复制当前点击的节点，并将它追加到《ul》元素中，当添加参数时复制它的事件   
  
    });   
    \*/   
    //替换节点   
   // $("p").replaceWith("<strong>你最不喜欢的水果是？</Strong>");     
  
   });   
</script>   
</head>   
<body>   
   <p>你好！</p>   
   你最喜欢的水果是？   
   <ul>   
    <li title="苹果">苹果</li>   
    <li title="橘子">橘子</li>   
    <li title="菠萝">菠萝</li>   
    </ul>   
</body>   
</html>

66、写一个继承方法？

function Parent(firstname) {  
    this.fname = firstname;  
    this.age = 40;  
    this.sayAge = function () {  
        console.log(this.age);  
    }  
}  
function Child(firstname) {  
    this.parent = Parent;  
    this.parent(firstname);  
    delete this.parent;  
    this.saySomeThing = function () {  
        console.log(this.fname);  
        this.sayA

67、写出this指向的几种情况？

68、XTMHTTPRequest的readystate的五种状态？

0)未初始化  
此阶段确认XMLHttpRequest对象是否创建，并为调用open()方法进行未初始化作好准备。值为0表示对象已经存在，否则浏览器会报错－－对象不存在。  
(1)载入  
此阶段对XMLHttpRequest对象进行初始化，即调用open()方法，根据参数(method,url,true)完成对象状态的设置。并调用send()方法开始向服务端发送请求。值为1表示正在向服务端发送请求。  
(2)载入完成  
此阶段接收服务器端的响应数据。但获得的还只是服务端响应的原始数据，并不能直接在客户端使用。值为2表示已经接收完全部响应数据。并为下一阶段对数据解析作好准备。  
(3)交互  
此阶段解析接收到的服务器端响应数据。即根据服务器端响应头部返回的MIME类型把数据转换成能通过responseBody、responseText或responseXML属性存取的格式，为在客户端调用作好准备。状态3表示正在解析数据。  
(4)完成  
此阶段确认全部数据都已经解析为客户端可用的格式，解析已经完成。值为4表示数据解析完毕，可以通过XMLHttpRequest对象的相应属性取得数据。  
概而括之，整个XMLHttpRequest对象的生命周期应该包含如下阶段：  
创建－初始化请求－发送请求－接收数据－解析数据－完成

69、status返回的窗口状态栏文本类型？（或http的状态码？）

70、写一个ajax使用get请求数据的全套代码？

<script type="text/javascript">  
    /\* 创建 XMLHttpRequest 对象 \*/  
    var xmlHttp;  
    function GetXmlHttpObject(){  
        if (window.XMLHttpRequest){  
            // code for IE7+, Firefox, Chrome, Opera, Safari   
            xmlhttp=new XMLHttpRequest();  
        }else{// code for IE6, IE5   
            xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");  
        }  
        return xmlhttp;  
    }  
    // -----------ajax方法-----------//   
    function getLabelsGet(){  
        xmlHttp=GetXmlHttpObject();  
        if (xmlHttp==null){  
            alert('您的浏览器不支持AJAX！');  
            return;  
        }  
        var id = document.getElementById('id').value;  
        var url="C:\Users\ADMINI~1\AppData\Local\Temp\%W@GJ$ACOF(TYDYECOKVDYB.pnghttp://www.Leefrom.com?id="+id+"&t/"+Math.random();  
        xmlHttp.open("GET",url,true);  
        xmlHttp.onreadystatechange=favorOK;//发送事件后，收到信息了调用函数   
        xmlHttp.send();  
    }  
    function getOkGet(){  
        if(xmlHttp.readyState==1||xmlHttp.readyState==2||xmlHttp.readyState==3){  
            // 本地提示：加载中   
        }  
        if (xmlHttp.readyState==4 && xmlHttp.status==200){  
            var d= xmlHttp.responseText;  
            // 处理返回结果   
        }  
    }  
</script>

71、URL参数截取？

function getvl(name) {

var reg = new RegExp("(^|\\?|&)"+ name +"=([^&]\*)(\\s|&|$)", "i");

if (reg.test(location.href)) return unescape(RegExp.$2.replace(/\+/g, " "));

return "";

};

<script>

getvl(bb);//调用传入需要获取的参数

</script>

72、js中如何对特殊符号进行转码与编码。

js对文字进行编码涉及3个函数：escape,encodeURI,encodeURIComponent，相应3个解码函数：unescape,decodeURI,decodeURIComponent  
1、   传递参数时需要使用encodeURIComponent，这样组合的url才不会被#等特殊字符截断。                              
例如：<script language="javascript">document.write('<a href="C:\Users\ADMINI~1\AppData\Local\Temp\8LDO48C$8@[GWU0353$FOVS.pnghttp://passport.baidu.com/?logout&aid=7&u= +encodeURIComponent("C:\Users\ADMINI~1\AppData\Local\Temp\8LDO48C$8@[GWU0353$FOVS.pnghttp://cang.baidu.com/bruce42")+">退出</a>');</script>  
2、   进行url跳转时可以整体使用encodeURI  
例如：Location.href=encodeURI("C:\Users\ADMINI~1\AppData\Local\Temp\8LDO48C$8@[GWU0353$FOVS.pnghttp://cang.baidu.com/do/s?word=百度&ct=21");  
3、   js使用数据时可以使用escape  
例如：搜藏中history纪录。  
4、   escape对0-255以外的unicode值进行编码时输出%u\*\*\*\*格式，其它情况下escape，encodeURI，encodeURIComponent编码结果相同。  
最多使用的应为encodeURIComponent，它是将中文、韩文等特殊字符转换成utf-8格式的url编码，所以如果给后台传递参数需要使用encodeURIComponent时需要后台解码对utf-8支持（form中的编码方式和当前页面编码方式相同）  
escape不编码字符有69个：\*，+，-，.，/，@，\_，0-9，a-z，A-Z  
encodeURI不编码字符有82个：!，#，$，&，'，(，)，\*，+，,，-，.，/，:，;，=，?，@，\_，~，0-9，a-z，A-Z  
encodeURIComponent不编码字符有71个：!， '，(，)，\*，-，.，\_，~，0-9，a-z，A-Z

73、在什么情况下你会用ajax请求来代替通常的跳转。

74、谈谈你对跨域访问的认识，写出两个实现跨域访问页面的的方法。

有一种简明的说法来解释广域跨域：跨域访问，简单来说就是 A 网站的 JavaScript 代码试图访问 B 网站，包括提交内容和获取内容。由于安全原因，跨域访问是被各大浏览器所默认禁止的。  
参考理论一：在浏览器中不能直接来跨域访问，而在服务器端没有跨域安全限制。  
这样的话，可以在服务端完成跨域访问，而在客户端来取得结果就可以了。

75、列出你所知道的设计模式并简述。

1. 1. 单例模式(singleton) ...
2. 2. 简单工厂(StaticFactory Method) ...
3. 3. 工厂方法(Factory Method)和抽象工厂(Abstract Factory) ...

76、javascr中有哪些数据类型；

字符串、数字、布尔、数组、对象、Null、Undefined

77、截取字符串abcde中的cde；

<**script**>  
 var ***str*** = "a,b,c,d,e";  
 alert(***str***.substring(3))  
</**script**>

78、什么是闭包，请举例说明；

闭包：函数内部返回一个函数，返回出来的函数被我们称之为闭包

var test = (function () {

console.log("test func");

var i = 0;

return function () {

console.log(i);

return i++;

}

})();

console.log("........");

test(); //0

test(); //1

test(); //2

// output

// test func

// ........

// 0

// 1

// 2

79、一个dom节点被点击的时候，我们希望能够执行一个函数，应该怎么做；

80、请写一个javascript函数parseQueryString，他的用途是吧url参数解析为一个对象，如：

var url = <http://www.test.com/index.php?key0=0&key1=1&key2=2>

var url = "C:\Users\ADMINI~1\AppData\Local\Temp\%W@GJ$ACOF(TYDYECOKVDYB.pnghttp://www.test.com/index.php?key0=0&key1=1&key2=2"  
var url = "C:\Users\ADMINI~1\AppData\Local\Temp\%W@GJ$ACOF(TYDYECOKVDYB.pnghttp://www.test.com/index.php?key0=0&key1=1&key2=2"  
function parseQueryString(str){  
    var param = {};  
  
    if(!str || ('string' !== typeof str && String !== str.constructor)){  
        return param;  
    }  
  
    var str = str.split("?");  
    if(str && str.length > 1){  
        str = str[1];  
    }else{  
        return {};  
    }  
  
    if(str){  
        str = str.split("&");  
    }  
  
    for(var index = 0; index < str.length; index ++){  
        var obj =  str[index].split('=');  
        param[obj[0]] = obj[1];  
    }  
    return param  
}  
parseQueryString('C:\Users\ADMINI~1\AppData\Local\Temp\%W@GJ$ACOF(TYDYECOKVDYB.pnghttp://www.test.com/index.php?key0=0&key1=1&key2=2');

81、怎样添加移除、移动、复制、创建和查找节点？

上边有

82、在javascript中什么是伪数组，如何将伪数组转化为标准数组；

伪数组（类数组）：  
典型的是函数的argument参数，还有像调用getElementsByTagName  
,document.childNodes之类的,它们都回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。  
在 JavaScript 中， 函数中的隐藏变量 arguments 和用 getElementsByTagName 获得的元素集合（NodeList）都不是真正的数组，不能使用 push 等方法，在有这种需要的时候只能先转换为真正的数组。  
对于 arguments，可以使用 Array.prototype.slice.call(arguments); 来达到转换的目的，但对于 NodeList 则不行了，其在 IE8 及以下将会报错，只能说其 JS 引擎有所限制。  
因此，如果需要把 NodeList 转换为真正的数组，则需要做下兼容处理。  
function makeArray(c) {  
    try  
    {  
        return  
  
        Array.prototype.slice.call(c);  
    }  
    catch  
            (e) {  
        var  
                ret = [], i = 0, len = c.length;  
        for  
        (; i < len; i++) {  
            ret[i]  
                    = (c[i]);  
        }  
        return  
        ret;  
    }  
}  
以下分别测试以上三种伪数组：  
//定义一个函数fun,内部使用makeArray将其arguments转换成数组  
function fun(){  
    var ary = makeArray(arguments);  
    alert(ary.constructor );  
}  
//调用  
fun(3,5);  
  
//假设页面上有多个段落元素p  
var els = document.getElementsByTagName("p");  
var ary1 = makeArray(els);  
alert(ary1.constructor);  
  
//特殊的js对象（如jquery对象）  
var obj={};  
obj[0] = "一";  
obj[1] = "二";  
obj[2] = "三";  
obj.length = 3;  
  
var ary2 = makeArray(obj);  
alert(ary2.constructor);

83、手写数组快速排序；

84、统计字符号串“aaaabbbccccddfggh”中字母个数或统计最多字母数；

//统计字符串中字母个数或统计最多字母数。  
var str = "aaaabbbccccddfggh";  
var obj  = {};  
for(var i=0;i<str.length;i++){  
    var v = str.charAt(i);  
    if(obj[v] && obj[v].value == v){  
        obj[v].count = ++ obj[v].count;  
    }else{  
        obj[v] = {};  
        obj[v].count = 1;  
        obj[v].value = v;  
    }  
}  
for(key in obj){  
    document.write(obj[key].value +'='+obj[key].count+'&nbsp;'); // a=4  b=3  c=4  d=2  f=1  g=2  h=1

85、写一个function，清楚字符串前后的空格（兼容所有浏览器）

1. //数组实现快速排序
2. #include <iostream>
3. **using** **namespace** std;
5. **void** sort(**int** array[],**int** X,**int** Y)
6. {
7. **int** z,y,i,k;
9. **if**(X<Y)
10. {
11. z=X;
12. y=Y;
13. k=array[z]; //key值
15. **do**  {
16. **while**((z<y)&&(array[y]>=k))
17. y--;
18. **if**(z<y)          //右边的元素小于k，移到k左
19. {
20. array[z]=array[y];
21. z=z+1;
22. }
23. **while**((z<y)&&(array[z])<=k)
24. z++;
25. **if**(z<y)             //左边的元素大于k，移动右边
26. {
27. array[y]=array[z];
28. }
30. } **while**(z!=y);
31. array[z]=k;//复制key值到array[z]
33. //  for(i=X;i<=Y;i++)
34. //             {
35. //              cout<<"a["<<i<<"]="<<array[i]<<";";
36. //               }
37. //      cout<<endl;
38. sort(array,X,z-1);// 对z的左侧进行sort操作
39. sort(array,z+1,Y);// 对z的右侧进行sort操作
40. }
41. }
42. **int** main()
43. {
44. //int a[10],i;
45. //  cout<<"请输入10个数"<<endl;
46. **char** A;
47. **do**
48. {
49. **int** **const** N=100;
50. **int** a[N];
51. **int** n;
52. cout<<"请输入需要排序的数字个数： ";
53. cin >> n;
54. cout<<"请输入 "<< n <<" 个数字:";
56. **for**(**int** i=0;i<n;i++)
57. { cin>>a[i];//输入
58. }
59. sort(a,0,n-1);//调用排序函数sort
60. cout<<"快速排序结果为";
62. **for** (i=0;i<n;i++)
63. cout<<a[i]<<"  ";//输出排序后的结果
65. cout<<"是否继续(y/n):"<<endl;
66. cin>>A;
67. }
68. **while**(A!='n' && A!='N');
69. **return** 0;
70. }

86、实现一个函数clone，可以对javascript中5种主要的数据类型（包括number string object array boolean）进行值复制；

array boolean）进行值复制；  
function clone(obj){  
var o;  
switch (typeof obj){  
case "undefiend":  
   break;  
case "string":  
 o = obj+"";  
 break;  
case "number":  
o = obj-0;  
break;  
case "boolean":  
o = obj;  
break;  
case "object":  
if (obj ===null){  
o = null;  
}else {  
if (Object.prototype.toString.call(obj).slice(8,-1)==="Array"){  
o = [];  
for (var i= 0;i<obj.length;i++){  
o.push(clone(obj[i]));  
}  
}els {  
o = {};  
for (var k in obj){  
o[k] = clone(obj[k]);  
}  
}  
}  
break;  
default:  
o = obj;  
break;  
}  
return o;  
}

87、将数组【“a”，“b”，“c”】转化为字符串；

88、获取enent目标对象的方法（要求兼容性）；

IE: 可以直接用window.event或event获得event对象,同时可直接用event.x或event.y获得当前操作位置的x和y坐标,当然还有其它属性,如:srcElement,keyCode等  
Firefox:不能直接使用window.event和event对象,当然也就不能获得该对象的其它属性.Firefox下获得操作位置坐标的属性是:pageX和pageY;   
下面提供同时兼容IE和Firefox的获得event对象的方法,在需要用到event对象的地方,调用该方法即可.  
function getEvent() //同时兼容ie和ff的写法   
    {    
        if(document.all)  return window.event;      
        func=getEvent.caller;          
        while(func!=null){    
            var arg0=func.arguments[0];   
            if(arg0)   
            {   
              if((arg0.constructor==Event || arg0.constructor ==MouseEvent) || (typeof(arg0)=="object" && arg0.preventDefault && arg0.stopPropagation))  
              {    
              return arg0;   
              }   
            }   
            func=func.caller;   
        }   
        return null;   
    }   
  
如调用:function test() {   
        var event=getEvent();   
        if (navigator.appName=='Microsoft Internet Explorer') {   
            alert(event.x);   
        }else if (navigator.appName=='Netscape') {   
            alert(event.pageX);   
        }   
          
    }

89、介绍js基本数据类型；

Undefined、Null、Boolean、Number和String。

90、null undefined 的区别；

Undefined类型只有一个值，即undefined。当声明的变量还未被初始化时，变量的默认值为undefined。  
Null类型也只有一个值，即null。null用来表示尚未存在的对象，常用来表示函数企图返回一个不存在的对象。

91、javascript中“use strict”是什么意思，为什么要使用；

1. "use strict" 表示的是按照“严格模式”执行[js代码](https://www.baidu.com/s?wd=js%E4%BB%A3%E7%A0%81&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YzmHI-nvwWujw9m1TLm1nv0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPH6Ln1Tvrjcdn16YP1mkPj6d" \t "_blank)，详细解释可以去看文档[http://www.ruanyifeng.com/blog/2013/01/javascript\_strict\_mode.html](http://www.ruanyifeng.com/blog/2013/01/javascript_strict_mode.html" \t "_blank)
2. 这是一个[三元运算符](https://www.baidu.com/s?wd=%E4%B8%89%E5%85%83%E8%BF%90%E7%AE%97%E7%AC%A6&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YzmHI-nvwWujw9m1TLm1nv0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPH6Ln1Tvrjcdn16YP1mkPj6d" \t "_blank)，语法是 条件 ? 结果1 : 结果2，满足条件时结果1否则结果2。

        具体释义： 如果a[f]存在， 则令 a[f] = a[f]； 否则，则执行括号里面的语句

92、描述一下javascript 的作用域链；

任何一段JavaScript代码都对应一个作用域链，作用域链中存放一系列对象，代码中声明的变量将作为对象的属性存放。  
在JavaScript的顶层代码中，作用域链由一个全局对象组成；当定义一个函数时，它保存一个作用域链，作用域链上有两个对象，一个是函数对象，一个是全局对象。  
每当一个函数被调用时，会创建一个活动对象（也叫上下文对象），函数中的局部变量将作为该对象的属性存放。  
当需要使用一个变量时，将从作用域链中逐个查找对象的属性。比如：要使用变量a，将先查找作用域中的第一个对象是否有属性a，如果有就使用；如果没有就查找作用域链中下一个对象的属性，以此类推。如果作用域链上没有任何一个对象含有属性 x，则认为这段代码的作用域链上不存在x，将抛出引用错误异常。  
当函数调用完成后，如果没有其他引用指向为此次调用所创建的上下文对象，该对象将被回收。  
var a = 100; function f() { var a = 200; function g() { return a; } return g; } console.log(f()());  
上述代码运行，会输出200。   
上述代码意味着，先执行 f()，得到该函数内嵌套的一个函数对象 g，然后调用这个嵌套的函数。   
定义函数 f 时，它保存一个作用域链，作用域链上有两个对象，一个是函数对象，一个是全局对象，此时，全局变量 a 是全局对象的属性，和函数对象 f 无关。   
执行函数 f 时，会创建一个活动对象，其中保存变量 a 作为函数 f 的属性而存在。而函数g嵌套定义在函数 f里，对其访问变量 a时，它没有变量 a，则继续往下查找函数 f。找到函数 f 的属性 a 并返回。因此，输出结果为 200。   
这种函数将变量包裹起来，外部代码可以通过内嵌函数 g来访问函数 f 内的局部变量的方式，也称为闭包。

93、什么是闭包，为什么要使用它；

指的是一个拥有许多变量和绑定了这些变量的环境的表达式（通常是一个函数），因而这些变量也是该表达式的一部分。

为了设置私有属性，保护闭包内数据不被干扰

94、谈谈对this对象的理解；

1：this谁调用指向谁  
2.闭包中的this指向window  
3.setInterval和setTimeout中的this指向window  
4.事件中的this  
        HTML：this指向window  
        DOMO：this指向绑定该事件的元素  
        DOM2级：IE中指向window，非IE中指向绑定该事件的元素  
5.自执行函数this指向window  
6.可以借助call和apply来改变this指向

95、判断一个字符串中各个字符出现的次数；例如字符串：hello 处理后：h：1 e：1 l:2 0:1

96、编写一个方法，判断字符串是否是这样组成的，第一个必须为数字，最后一个必须为 字母  
  
字母  
  function isWant(str){  
            var reg = /^\d[\d\D]\*[a-zA-Z]$/;  
            return reg.test(str);  
        }

97、编写一个方法，出去数组中的重复的元素：例如数字【“ab”，“ab”，“ac”，“cd”】 处 理后、：【“ab”，“ac”，“cd”】

98、如何隐藏一个dom元素

设置css display属性为none，效果：元素不显示，不占位  
设置css visibility属性为hidden，效果：元素不显示，但占位

99、javascript 中有哪些基础数据类型

Undefined、Null、Boolean、Number和String。

100、如何添加HTML事件，有几种方法？（至少两种方式）

1. 通过HTML元素属性。简单说来就是在html结构中，给你要添加事件的元素添加一个属性。  
属性名为 'on' + 事件名。  
如:你要给a元素绑定一个click事件，你就该这么写：  
<a href="C:\Users\ADMINI~1\AppData\Local\Temp\[5UQ[BL(6~BS2JV6W}N6[%S.pnghttp://blog.163.com/luping\_01/blog/#" onclick='do something'>name</a>  
       2.通过对象属性。  
对象指的是jDOM树里的对象，我们都知道，所有的html元素在DOM(文档对象类型)里都存在一个相应的DOM元素。  
给这个DOM元素添加事件等同于方式一。这个属性名也一样：  
也为 on + 事件名  
如：  
html结构:  
<a href="C:\Users\ADMINI~1\AppData\Local\Temp\[5UQ[BL(6~BS2JV6W}N6[%S.pnghttp://blog.163.com/luping\_01/blog/#" id="n">name</a>  
js：  
document.getElementById('n').onclick = function(){ //do something}  
     3.通过W3C监听方式（标准方式）或者IE专属的中间模型添加事件  
W3C方式： elemennt.addEventListener(事件名,处理函数引用,true || false)  
这里的事件名就是单纯的事件名，不需要加 on ，true表示在事件传播阶段捕获事件，false表示在事件冒泡阶段捕获事件，一般我们设为false!  
IE模式: element.attachEvent( 'on' + 事件名，处理函数引用);  
注意，IE模式需要加on

101、javascript 面向对象中继承实现方式

**一、原型链继承**

**二、使用apply、call方法**

**三、对象实例间的继承**

102、javascript 中如何让对一个对象进行深拷贝？

方法一：使用 JSON.parse() 方法  
要实现深拷贝有很多办法，比如最简单的办法是使用 JSON.parse()：  
/\* ================ 深拷贝 ================ \*/  
function deepClone(initalObj) {  
    var obj = {};  
    try {  
        obj = JSON.parse(JSON.stringify(initalObj));  
    }  
    return obj;  
}  
  
/\* ================ 客户端调用 ================ \*/  
var obj = {  
    a: {  
        a: "world",  
        b: 21  
    }  
}  
var cloneObj = deepClone(obj);  
cloneObj.a.a = "changed";  
    
console.log(obj.a.a); // "world"

103、为什么异步加载js文件？加载方式？（至少两种方式）

方法一，jQuery.getScript

HTML 代码：

代码如下 复制代码   
<button id="go">Run</button>  
<div class="block"></div>

jQuery 代码：

jQuery.getScript("http://dev.jquery.com/view/trunk/plugins/color/jquery.color.js",  
function(){  
$("#go").click(function(){  
$(".block").animate( { backgroundColor: 'pink' }, 1000)  
.animate( { backgroundColor: 'blue' }, 1000);  
});  
});

104、前端常规开发优化策略（至少两种方式）

1、尽量减少HTTP请求个数——须权衡        1  
2、为文件头指定Expires或Cache-Control，使内容具有缓存性。     
3、避免空的src和href       
4、使用gzip压缩内容        
5、把CSS放到顶部       
6、把JS放到底部          
7、避免使用CSS表达式  
8、将CSS和JS放到外部文件中        3  
9、精简CSS和JS        3  
10、剔除重复的JS和CSS(代码的复用性)        3  
11、使AJAX可缓存        3  
12、尽早刷新输出缓冲        4  
13、使用GET来完成AJAX请求        4  
14、延迟加载        4  
15、预加载        4  
16、尽量减少iframe的个数        4  
17、避免404        5  
18、减少Cookie的大小        5  
19、优化图像        5  
20、不要在HTML中缩放图像——须权衡        5  
21、使用CDN（内容分发网络）        5  
22、避免跳转        6  
23、配置ETags        6  
24、减少DOM访问        6  
25、开发智能事件处理程序        6  
26、用<link>代替@import        6  
27、避免使用滤镜        7  
28、favicon.ico要小而且可缓存        7  
29、保持单个内容小于25K        7  
30、打包组件成复合文本        7

105、如何控制网页在网络传输过程中的数据量

106、var stringArray = [ "this","is","chunchujie"]，请在控制台输出“this is chunchujie”。

console.log(stringArray.join(' '));

107、请写一个函数去除数组的重复项、；如[2008,34,78,2,45,34,78,90,90,12,34]处理后为[20 0 8.34，78,2,45,90,12]

<**script**>  
 Array.**prototype**.deleteData = function(){  
 var arr=[];  
 var deletearr = {};  
 for (var i = 0; i < this.**length**; i++) {  
 if (!deletearr[this[i]]) {  
 arr.push(this[i]);  
 deletearr[this[i]] = 1  
 }  
 }  
 return arr;  
 };  
 var ***arr*** = [2008,34,78,2,45,34,78,90,90,12,34];  
 alert(***arr***.deleteData())  
</**script**>

108、用js或jQuery获取id为chuchujie的div里边的所有a标签，并把字体大小设置为18px；同 时把第三个a标签设置为块级元素，并把他的宽高分别设置为300px和500px；

var as = document.getElementById('chuanchuanjie').getElementsByTagName('a');  
        for(var i = 0,len = as.length;i<len;i++){  
            if(i==2){  
                as[i].style.display = 'block';  
                as[i].style.height = '500px';  
                as[i].style.width = '300px';  
            }  
            as[i].style.fontSize = '18px';  
        }

109、请在网页上输出九九乘法表；

<**script**>  
 for (var ***i*** = 1; ***i*** < 10; ***i***++) {  
 for (var ***j*** = ***i***; ***j*** > 0; ***j***--) {  
 **document**.write(***i*** + "\*" + ***j*** + "=" + ***i*** \* ***j***);  
 **document**.write(" ")  
 }  
 **document**.write("<br/>")  
 }  
</**script**>

110、call 和apply的区别是什么？

call方法:   
语法：call(thisObj，Object)  
定义：调用一个对象的一个方法，以另一个对象替换当前对象。  
说明：  
call 方法可以用来代替另一个对象调用一个方法。call 方法可将一个函数的对象上下文从初始的上下文改变为由 thisObj 指定的新对象。   
如果没有提供 thisObj 参数，那么 Global 对象被用作 thisObj。   
  
apply方法：   
语法：apply(thisObj，[argArray])  
定义：应用某一对象的一个方法，用另一个对象替换当前对象。   
说明：   
如果 argArray 不是一个有效的数组或者不是 arguments 对象，那么将导致一个 TypeError。   
如果没有提供 argArray 和 thisObj 任何一个参数，那么 Global 对象将被用作 thisObj， 并且无法被传递任何参数。

111、请解释jsonp的工作原理

首先在客户端注册一个callback, 然后把callback的名字传给服务器。

此时，服务器先生成 json 数据。

然后以 javascript 语法的方式，生成一个function , function 名字就是传递上来的参数 jsonp.

最后将 json 数据直接以入参的方式，放置到 function 中，这样就生成了一段 js 语法的文档，返回给客户端。

客户端浏览器，解析script标签，并执行返回的 javascript 文档，此时数据作为参数，传入到了客户端预先定义好的 callback 函数里.（动态执行回调函数）

112、请简要说明javascript 中this 是如何让工作的的

call方法:   
语法：call(thisObj，Object)  
定义：调用一个对象的一个方法，以另一个对象替换当前对象。  
说明：  
call 方法可以用来代替另一个对象调用一个方法。call 方法可将一个函数的对象上下文从初始的上下文改变为由 thisObj 指定的新对象。   
如果没有提供 thisObj 参数，那么 Global 对象被用作 thisObj。   
  
apply方法：   
语法：apply(thisObj，[argArray])  
定义：应用某一对象的一个方法，用另一个对象替换当前对象。   
说明：   
如果 argArray 不是一个有效的数组或者不是 arguments 对象，那么将导致一个 TypeError。   
如果没有提供 argArray 和 thisObj 任何一个参数，那么 Global 对象将被用作 thisObj， 并且无法被传递任何参数。

113、请谈谈你都使用过哪些javascript模板

114、你怎么看AMD vs.Commonjs?

new foo() // 输出： {} 在构造函数内部创建一个对象

113、请谈谈你都使用过哪些javascript模板

114、你怎么看AMD vs.Commonjs?

js开发者对js模块加载的尝试和创新从来都没有停止过，尤其是当nodejs的出现后，模块化加载的必要性更加凸显。本文不讨论如何在nodejs环境来模块化加载（创造者已经利用commonJS机制解决），只讨论在浏览器环境下如何来模块加载的思路，并提出一些我的看法。

浏览器环境与nodejs的环境的最大差异是，对于nodejs的环境，大多数情况下被依赖的模块文件本身就在本地（它们都在服务器上），同步取过来就能用；而对于浏览器的环境，被依赖的模块文件通常还在远程服务器上，并未加载到本地，也就是说必须是先加载（并解析）后执行的机制。

既然已经有了commonJS，在这之上将异步回调的逻辑加入进去，可是异步先加载什么呢？于是就有了依赖的概念。一段时间的发展后，有了AMD、和CMD的解决方案，代表作品是requirejs和seajs，有兴趣的读者可以去了解一下，这里就不展开介绍了。

有必要简单提一下两者的主要区别，CMD推崇依赖就近，可以把依赖写进你的代码中的任意一行，例：

define(function(require, exports, module) {

var a = require('./a')

a.doSomething()

var b = require('./b')

b.doSomething()

})

代码在运行时，首先是不知道依赖的，需要遍历所有的require关键字，找出后面的依赖。具体做法是将function toString后，用正则匹配出require关键字后面的依赖。显然，这是一种牺牲性能来换取更多开发便利的方法。

而AMD是依赖前置的，换句话说，在解析和执行当前模块之前，模块作者必须指明当前模块所依赖的模块，表现在require函数的调用结构上为：

define(['./a','./b'],function(a,b){

a.doSomething()

b.doSomething()

})

代码在一旦运行到此处，能立即知晓依赖。而无需遍历整个函数体找到它的依赖，因此性能有所提升，缺点就是开发者必须显式得指明依赖——这会使得开发工作量变大，比如：当你写到函数体内部几百上千行的时候，忽然发现需要增加一个依赖，你不得不回到函数顶端来将这个依赖添加进数组。

细心的读者可能发现，到目前位置我讨论的AMD和CMD的思想的关于依赖的部分，都只讨论的“硬依赖”，也就是执行前肯定需要的依赖，但是这不是全部的情况。有的时候情况是这样的：

// 函数体内：

if(status){

a.doSomething()

}

在这个函数体内，可能依赖a，也可能不依赖a，我把这种可能的依赖成为“软依赖”。对于软依赖当然可以直接当硬依赖处理，但是这样不经济，因为依赖是不一定的，有可能加载了此处的依赖而实际上没有用上。

对于软依赖的处理，我推荐依赖前置+回调函数的实现形式。上面的例子简单表述如下：

// 函数体内：

if(status){

async(['a'],function(a){

a.doSomething()

})

}

至此可以对由commonJS衍生出来的方案做出总结了。在浏览器端来设计模块加载机制，需要考虑依赖的问题。

我们先把依赖分为两种，“强依赖” —— 肯定需要 和“弱依赖” —— 可能需要。

对于强依赖，如果要性能优先，则考虑参照依赖前置的思想设计你的模块加载器，我个人也更推崇这个方案一些；如果考虑开发成本优先，则考虑按照依赖就近的思想设计你的模块加载器。

对于弱依赖，只需要将弱依赖的部分改写到回调函数内即可。

如果现在我要实现一个模块加载器，我会将强依赖前置，弱依赖采用异步回调函数的形式，其它的方法我认为都只是语法糖而已，仅此就够了。

115、使用Promises 而非回调（callbacks）优缺点是什么？

116、如何实现下列代码：

var number =[ 1,2,3,4,5];

number.square();//[1,4,9,16,25]

number.average();//3

number.sum();//15

number.even()://[2,4]

Array.prototype.square = function () {  
        var arr = [];  
        for(var i=0;i<this.length;i++){  
            arr[i] = this[i]\*this[i];  
        }  
        console.log(arr);  
        return arr;  
    };  
    Array.prototype.average= function () {  
        var jieguo=0;  
        for(var i=0;i<this.length;i++){  
            jieguo = jieguo+this[i];  
        }  
        console.log(jieguo/this.length);  
        return jieguo/this.length;  
    };  
    Array.prototype.sum= function () {  
        var jieguo=0;  
        for(var i=0;i<this.length;i++){  
            jieguo = jieguo+this[i];  
        }  
        console.log(jieguo);  
        return jieguo;  
    };  
    Array.prototype.even= function () {  
        var arr=[];  
        for(var i=0;i<this.length;i++){  
            if(this[i]%2==0){  
                arr.push(this[i]);  
            }  
        }  
        console.log(arr);  
        return arr;  
    };  
  
    number.square();  
    number.average();  
    number.sum();  
    number.even();  
  
  
  
   // var number = [1, 2, 3, 4, 5];  
        // number.square();//[1,4,9,16,25]  
        // number.average();//3  
        // number.sum();//15  
        // number.even()://[2,4]  
        var number = [1, 2, 3, 4, 5];  
        Array.prototype.square = function(){  
            var result = this.map(function(value,index,arr){  
                return value\*value;  
            });  
            return result;  
        };  
        Array.prototype.average = function(){  
            var sum = 0;  
            this.forEach(function(value,index,arr){  
                sum+=value;  
            });  
            return sum/this.length;  
        };  
         Array.prototype.sum = function(){  
            var sum = 0;  
            this.forEach(function(value,index,arr){  
                sum+=value;  
            });  
            return sum;  
        };  
        Array.prototype.even = function(){  
            var result = this.filter(function(value,index,arr){  
                return value%2==0;  
            });  
            return result;  
        };

117、请实现一个为数组去重的函数

<**script**>  
 Array.**prototype**.deleteData = function(){  
 var arr=[];  
 var deletearr = {};  
 for (var i = 0; i < this.**length**; i++) {  
 if (!deletearr[this[i]]) {  
 arr.push(this[i]);  
 deletearr[this[i]] = 1  
 }  
 }  
 return arr;  
 };  
 var ***arr*** = [2008,34,78,2,45,34,78,90,90,12,34];  
 alert(***arr***.deleteData())  
</**script**>

118、请使用任意一种mvvm框架演示如何实现双向绑定

<div id="container">  
  <p>  
  name:<input type="text" bindkey="userName">  
  </p>  
  <p>  
  age:<input type="text" bindkey="age">  
  </p>  
<div>  
  
<script type="text/javascript">  
  window.Model={  
    userName:"windy",  
    age:34,  
    skill:["javascript","html","css","jquery","node"],  
       
  }  
  function bindingModel(model,changeCallback){  
    var propertiesMap={};  
    model.\_\_private={};  
    function getFn(name){  
      var result=this.\_\_private[name]  
      console.log("get value:"+name+"="+ result);  
      return result;  
    };  
    function setFn(name,val){  
      if(this.\_\_private[name]!=val){  
        console.log("set value:"+name+"="+val);  
          
        this.\_\_private[name]=val;  
   
        if(changeCallback){  
          changeCallback(name,val);  
        }  
      }  
    };  
    for(elem in model){  
      if(model.hasOwnProperty(elem) && elem!="\_\_private" && typeof(model[elem])!="function"){  
        (function(propName,propValue){  
          model.\_\_private[propName]=propValue;// init value  
          propertiesMap[propName]={  
            get:function(){ return getFn.call(this,propName)},  
            set:function(v){ return setFn.call(this,propName,v)},  
            //value:model[elem],  
            //writable: true,  
            enumerable: true,  
            configurable: true  
          }  
        })(elem,model[elem]);  
      }  
    }  
    Object.defineProperties(model,propertiesMap)  
       
  }  
  function bindingBoth(model,dom){  
    dom.find("[bindkey]").each(function(item){  
      var key=$(this).attr("bindkey");  
      $(this).val(model[key]);  
      $(this).bind("input",function(){  
        model[key]=$(this).val();  
      })  
    });  
    bindingModel(model,function(name,val){  
      var el=dom.find("[bindkey="+name+"]");  
      if(el.val()!=val){  
        el.val(val);  
      }  
         
    });  
  }  
  bindingBoth(window.Model,$("#container"))  
  </script>

119、svg 与canvas的不同

**SVG**

SVG 是一种使用 XML 描述 2D 图形的语言。

SVG 基于 XML，这意味着 SVG DOM 中的每个元素都是可用的。您可以为某个元素附加 JavaScript 事件处理器。

在 SVG 中，每个被绘制的图形均被视为对象。如果 SVG 对象的属性发生变化，那么浏览器能够自动重现图形。

**Canvas**

Canvas 通过 JavaScript 来绘制 2D 图形。

Canvas 是逐像素进行渲染的。

在 canvas 中，一旦图形被绘制完成，它就不会继续得到浏览器的关注。如果其位置发生变化，那么整个场景也需要重新绘制，包括任何或许已被图形覆盖的对象。

### Canvas

* 依赖分辨率
* 不支持事件处理器
* 弱的文本渲染能力
* 能够以 .png 或 .jpg 格式保存结果图像
* 最适合图像密集型的游戏，其中的许多对象会被频繁重绘

### SVG

* 不依赖分辨率
* 支持事件处理器
* 最适合带有大型渲染区域的应用程序（比如谷歌地图）
* 复杂度高会减慢渲染速度（任何过度使用 DOM 的应用都不快）
* 不适合游戏应用

120、localhost与sessionStorage的不同

localStorage和sessionStorage一样都是用来存储客户端临时信息的对象。  
  
他们均只能存储字符串类型的对象（虽然规范中可以存储其他原生类型的对象，但是目前为止没有浏览器对其进行实现）。  
  
localStorage生命周期是永久，这意味着除非用户显示在浏览器提供的UI上清除localStorage信息，否则这些信息将永远存在。  
  
sessionStorage生命周期为当前窗口或标签页，一旦窗口或标签页被永久关闭了，那么所有通过sessionStorage存储的数据也就被清空了。  
  
不同浏览器无法共享localStorage或sessionStorage中的信息。相同浏览器的不同页面间可以共享相同的 localStorage（页面属于相同域名和端口），但是不同页面或标签页间无法共享sessionStorage的信息。这里需要注意的是，页面及标 签页仅指顶级窗口，如果一个标签页包含多个iframe标签且他们属于同源页面，那么他们之间是可以共享sessionStorage的。  
  
同源的判断规则：  
  
C:\Users\ADMINI~1\AppData\Local\Temp\%W@GJ$ACOF(TYDYECOKVDYB.pnghttp://www.test.com  
  
C:\Users\ADMINI~1\AppData\Local\Temp\%W@GJ$ACOF(TYDYECOKVDYB.pnghttps://www.test.com （不同源，因为协议不同）  
  
C:\Users\ADMINI~1\AppData\Local\Temp\%W@GJ$ACOF(TYDYECOKVDYB.pnghttp://my.test.com（不同源，因为主机名不同）  
  
C:\Users\ADMINI~1\AppData\Local\Temp\%W@GJ$ACOF(TYDYECOKVDYB.pnghttp://www.test.com:8080（不同源，因为端口不同）  
  
localStorage和sessionStorage使用时使用相同的API：  
  
localStorage.setItem("key","value");//以“key”为名称存储一个值“value”  
  
localStorage.getItem("key");//获取名称为“key”的值  
  
枚举localStorage的方法：  
  
for(var i=0;i<localStorage.length;i++){  
  
     var name = localStorage.key(i)​;  
  
     var value = localStorage.getItem(name);​  
  
}  
  
删除localStorage中存储信息的方法：  
  
localStorage.removeItem("key");//删除名称为“key”的信息。  
  
localStorage.clear();​//清空localStorage中所有信息  
  
 通过getItem或直接使用localStorage["key"]获取到的信息均为实际存储的副本。  
  
例如：  
  
localStorage.key = {value1:"value1"}​;  
  
localStorage.key.value1='a'​;  
  
这里是无法​对实际存储的值产生作用的，下面的写法也不可以：  
  
​localStorage.getItem("key").value1="a";

122、jQuery Mobile 是什么

jQuery Mobile 是创建移动 web 应用程序的框架。  
Jquery mobile是一个基于HTML5，拥有响应式网站特性，兼容所有主流移动设备平台的统一UI接口系统，与前端开发框架。可以运行在所有智能手机，平板电脑和桌面设备上。 jQuery Mobile秉承 “write less, do more”的原则到了一个新的高度：不需要为每一个移动设备或者操作系统单独开发应用，jQuery Mobile框架可以使您设计一个高度响应式的网站或应用运行于所有流行的智能手机，平板电脑和桌面系统。

123、输出今天的日期，以YYYY-MM-DD的方式，比如今天是2016年10 月26日，则输出2016- 10-26.

124、怎样添加、移除、移动、复制、创建和查找结点（原生js）。

1、添加节点append方法  
栗子：为body添加一个内容为sss的文本节点  
document.body.appendChild(document.createTextNode("sss"));  
2、移除节点，removeChild移除某个节点的子节点  
栗子：  
var ccn=document.getElementById("sd").childNodes[0];//获取到要移除的节点  
    document.getElementById("sd").removeChild(ccn);//将id为sd的第一个子节点移除  
3、移动，控制  
栗子：将id为sd的节点向右边移动50px  
var sdds=document.getElementById("sd");  
    sdds.style.left=parseInt(sdds.style.left)+50+"px";  
<div id='sd' style="position: absolute; left: 10px;">sd</div>  
4、创建节点  
document.createTextNode("sss")//创建文本节点  
document.createElement("p")//创建p节点  
5、查找节点  
document.getElementById('oo');//根据id查找  
document.getElementsByTagName("p");//根据标签名字查找节点

125、原生JS的window.onload与jquery的$(document).ready(function(){}) 有什么不同？

window.onload是加载完结构和样式以后才加载js文件

126、想实现一个对页面某个结点的拖拽？如何做？（使用原生JS）。

<!DOCTYPE html>  
<html>  
<head>  
    <meta charset="utf-8">  
    <title>event事件</title>  
    <style type="text/css">  
        #box{width: 100px;height: 100px; background-color: red;position: absolute;}  
   
    </style>  
</head>  
<body>  
     <div id="box"></div>  
     <script type="text/javascript">  
          
       var box=document.getElementById('box');  
       var ww=document.documentElement.clientWidth;  
        var ww1=document.documentElement.clientHeight;  
        box.onmousedown=function (ev) {  
                var ev=ev||window.event;  
                var a=ev.clientX-box.offsetLeft;  
                var b=ev.clientY-box.offsetTop;  
                   
            document.onmousemove=function (ev) {  
                 var ev=ev||window.event;  
                 var c=ev.clientX-a;  
                 var d=ev.clientY-b;  
                     
                  if(c<=0){  
                      c=0;  
                  }  
                  //if (a>=ww-box.clientWidth) {  
                  //       c=ww-box.clientWidth;  
                  // }  
                     if (d<=0) {  
                       d=0;  
                  }  
                  // if (b>=ww1-box.clientHeight) {  
                  //       d=ww1-box.clientHeight;  
                  // }  
                    box.style.left=c+'px';  
                    box.style.top=d+'px';  
            }  
        }  
        box.onmouseup=function () {  
           document.onmousemove=null;  
        }  
                 
     </script>  
</body>  
<html>

127、JavaScript原型，原型链？

128、jquery与jQuery UI有什么区别？

jQuery UI 与 jquery 的主要区别是： 　  
　(1) jQuery是一个js库，主要提供的功能是选择器，属性修改和事件绑定等等。 　  
　(2) jQuery UI则是在jQuery的基础上，利用jQuery的扩展性，设计的插件。提供了一些常用的界面元素，诸如对话框、拖动行为、改变大小行为等等。 　  
　(3) jQuery本身注重于后台，没有漂亮的界面，而jQuery UI则补充了前者的不足，他提供了华丽的展示界面，使人更容易接受。既有强大的后台，又有华丽的前台。jQuery UI是jQuery插件，只不过专指由jQuery官方维护的UI方向的插件。

129、对Node的有点和缺点提出自己的看法？

130、一个div宽高100px，点击变大，当它的宽高等于400px开始缩小，当宽高缩小至100px

时， 开始放大；再次点击暂定缩小或放大，再再次点击，div继续放大或缩小。

var div = document.getElementById('div');  
        var isBig = true;  
        var isChange = false;  
        var timer = null;  
        function big(ele) {  
            if (ele.offsetWidth < 400) {  
                ele.style.width = ele.offsetWidth + 1 + 'px';  
                ele.style.height = ele.offsetHeight + 1 + 'px';  
                timer = setTimeout(function () {  
                    big(ele);  
                }, 10)  
            } else {  
                isBig = false;  
                small(ele);  
            }  
        }  
        function small(ele) {  
            if (ele.offsetWidth > 100) {  
                ele.style.width = ele.offsetWidth - 1 + 'px';  
                ele.style.height = ele.offsetHeight - 1 + 'px';  
                timer = setTimeout(function () {  
                    small(ele);  
                }, 10)  
            } else {  
                isBig = true;  
                big(ele);  
            }  
        }  
        div.onclick = function () {  
            if (isChange) {  
                clearTimeout(timer);  
                isChange = false;  
            } else {  
                if (isBig) {  
                    big(this);  
                } else {  
                    small(this);  
                }  
                isChange =true;  
            }  
  
        }

131、静态资源CDN如何使用。

132、用JS手写快速排序

quickSort = function(arr){  
    //返回（如果当前数组不再需要排序时）  
    if(arr.length <= 1) return arr;  
    //声明两个数组分别用来防止"小值"和"大值"  
    var less = [];  
    var greater = [];  
    //选取被排序数组中的任一元素作为"基准"（这里我们就选取数组中间的元素）  
    var pivotIndex = Math.floor(arr.length / 2);  
    var pivot = arr.splice(pivotIndex, 1)[0];  
    //遍历数组，进行区分操作  
    for(var i = 0, len = arr.length; i < len; i++){  
        if(arr[i] < pivot){  
            less.push(arr[i]);  
        } else {  
            greater.push(arr[i]);  
        }  
    }  
    //最后使用递归不断重复这个过程，直到获得排序后的数组  
    return quickSort(less).concat([pivot], quickSort(greater));  
};  
//实例  
var arr = [89,28,743,18,93,58,84,65];  
console.log("数组元素的原始排列：" + arr.join(","));  
console.log("排序后，数组元素的排列：" + quickSort(arr).join(","));

133、Postcss是啥

PostCSS 已经被许多大公司使用：谷歌，推特，阿里巴巴和 Shopify。 Autoprefixer PostCSS 插件是最流行的 CSS 处理器插件之一。  
PostCSS 可以作为预处理器使用，类似：Sass, Less 和 Stylus。但是 PostCSS 是模块化的工具，比之前那些快3-30 倍，而且功能更强大。  
PostCSS 包括 CSS 解析器，CSS 节点树 API，一个源映射生成器和一个节点树 stringifier。  
简而言之，PostCSS是CSS变成JavaScript的数据，使它变成可操作。PostCSS是基于JavaScript插件，然后执行代码操作。PostCSS自身并不会改变CSS，它只是一种插件，为执行任何的转变铺平道路。  
本质上是没有很制PostCSS插件操纵CSS，也就是说它可以适用于任何CSS。只要你能想到的，你都可以编写一个PostCSS插件，让它来转成CSS。  
PostCSS插件可以像预处理器，它们可以优化和autoprefix代码；可以添加未来语法；可以添加变量和逻辑；可以提供完整的网格系统；可以提供编码的快捷方式......还有很多很多。  
事实上，使用PostCSS插件你可以做任何你想做的任何事情，但PostCSS相对而言还是较新，这也导致了一些误解，工具实际上是什么。  
许多人（包括我自己在内）对PostCSS没有一个完整的印象，PostCSS实际是什么，并因此错过了PostCSS可以提供什么。

134、Java和Javascript的共同点

java是一门强类型的面向对象的编程语言,所有的对象都有类型,并且靠java虚拟机来解释执行  javascript虽然有java二字,实际和java关系不大,它是一门弱类型脚本语言,里面的变量申明的时候不带类型,只到解释执行的时候才确定类型。它是靠浏览器负责解释执行的  javascript可以被与很多语言如jsp php asp等一起形成丰富的网页  java分为三部分 ,不直接用于网页开发  j2se :用于开发桌面程序j2ee :用于企业分布式开发  j2me :用于移动平台如手机等开发

135、截取abcdefg的efg

<**script**>  
 var ***str*** = "abcdefg";  
 alert(***str***.substring(4))  
</**script**>

136、判断一个字符串中出现次数最多的字符，统计这个次数。

<p>判断一个字符串中出现次数最多的字符，统计这个次数</p>    
<button onclick="btn()">点击<button>    
<p id="p1"></p>    
<p id="p2"></p>    
<script>    
   var a=[0,1,2,1,2,5,5,3,1,3,5,1,2,4,0,5]   //给定数组    
    
   var p=document.getElementById('p1');    
   var p2=document.getElementById('p2');    
    var yuansu=new Array();   //储存元素    
    var count=new Array();    //储存元素个数    
    var m=1;    
    
   // 去重并统计重复的个数    
    function btn () {    
            for(i=0;i<a.length;i++){    
                for(x=i+1;x<a.length;x++){    
                    if(a[i]==a[x]){    
                        a.splice(x,1)    
                        m++;    
                    }    
                }    
               yuansu[i]=a[i];    
               count[i]=m;    
               m=1;    
            }    
    
    //对count数组 复制一份在进行排序    
    
        var coln=new Array();    
           for(s=0;s<count.length;s++){    
             coln[s]=count[s]    
           }    
           function abc (a,b) {    
               return a-b;    
           }    
           coln.sort(abc)    
    
    // 统计最多的元素 以及 有几个最多的    
           var fcount=0;  //有几个最多的    
           var first="";    
           for(i=0;i<count.length;i++){    
              if(count[i]==coln[coln.length-1]){    
               first+=yuansu[i]+"出现最多,出现"+count[i]+"次"+"<br>"    
                fcount++;    
              }    
    
           }    
    
         p.innerHTML="最多的有"+fcount+"个";    
         p2.innerHTML=first;    
    }    
    
    
    
   //算出出现次数居第二位的元素及个数    
   for (var i = 0; i < count.length; i++) {    
    
       if (count[i] == coln[coln.length - fcount]) {    
           //document.write("出现次数居第二位的元素是：" + yuansu[i] + "次数为：" + count[i] + "<br/>");    
           second += "出现次数居第二位的元素是：" + yuansu[i] + "次数为：" + count[i] + "<br/>"    
       }    
   }    
    
    
</script>

137、规避javascript多人开发函数重名问题

命名空间  
封闭空间  
js模块化mvc（数据层、表现层、控制层）  
seajs（如果了解的呢，可以说）  
变量转换成对象的属性  
对象化

138、Javascript面向对象中继承实现

call()方法：  
function useCall(a,b){  
  this.a = a;  
  this.b = b;  
  this.say=function(){  
    alert("I'm "+this.a+" You're"+this.b);  
  }  
 }  
 function callThefunction (){  
    var args = arguments;  
    useCall.call(this,args[0],args[1]);  
 //  useCall.apply(this,arguments);  
  }  
var testCall1 =new useCall("Not YY","Not TT");  
testCall1.say();  
var testCall2 = new callThefunction("YY","TT");  
testCall2.say();  
  
 原型方式：  
function car(price){  
 this.price=price;  
}  
car.prototype={  
    sayPrice:function(){console.log("Price is "+this.price);}  //console.log()参见Firebug API  
}  
function toyCar(price){  
 this.price=price;  
}  
toyCar.prototype=new car()  
var oCar=new car("100W");  
oCar.sayPrice();  
var oCar2=new toyCar("10CNY");  
oCar2.sayPrice();

139、编写一个方法，去掉一个数组的重复元素

* 上边有！！

140、Javascript中如何检测一个变量事一个string类型？请写出函数实现

function a(obj){

return typeof(obj)=="string";

}

alert(a(123));

alert(a("abc"));

141、请尽可能详尽的解释AJXA的工作原理

**ajax原理和XmlHttpRequest对象**

　　Ajax的原理简单来说通过XmlHttpRequest对象来向服务器发异步请求，从服务器获得数据，然后用javascript来操作DOM而更新页面。这其中最关键的一步就是从服务器获得请求数据。要清楚这个过程和原理，我们必须对 XMLHttpRequest有所了解。

　XMLHttpRequest是ajax的核心机制，它是在IE5中首先引入的，是一种支持异步请求的技术。简单的说，也就是javascript可以及时向服务器提出请求和处理响应，而不阻塞用户。达到无刷新的效果。

  　所以我们先从XMLHttpRequest讲起，来看看它的工作原理。

  　首先，我们先来看看XMLHttpRequest这个对象的属性。

  　　它的属性有：

  　　onreadystatechange  每次状态改变所触发事件的事件处理程序。

  　　responseText     从服务器进程返回数据的字符串形式。

  　　responseXML    从服务器进程返回的DOM兼容的文档数据对象。

  　　status           从服务器返回的数字代码，比如常见的404（未找到）和200（已就绪）

  　　status Text       伴随状态码的字符串信息

  　　readyState       对象状态值

　　　　0 (未初始化) 对象已建立，但是尚未初始化（尚未调用open方法）

　　　　1 (初始化) 对象已建立，尚未调用send方法

　　　　2 (发送数据) send方法已调用，但是当前的状态及http头未知

　　　　3 (数据传送中) 已接收部分数据，因为响应及http头不全，这时通过responseBody和responseText获取部分数据会出现错误，

　　　　4 (完成) 数据接收完毕,此时可以通过通过responseXml和responseText获取完整的回应数据

　　但是，由于各浏览器之间存在差异，所以创建一个XMLHttpRequest对象可能需要不同的方法。这个差异主要体现在IE和其它浏览器之间。下面是一个比较标准的创建XMLHttpRequest对象的方法。

142、 Function hi(){

Var a;

Alert(a);

}

结果：undefined

143、如下代码输出结果

Var b;

Function a(){

Var a=”a”;

B = function(){

Return a+”B”;

}

Return a;

}

Alert(a());  *a*

*Alert(B()); aB*

结果：

144、一下代码输出结果？

console.dir(0==false);true  
Console.dir(1==true);true  
Console.dir(2=={valueOf: function( ){return 2}});true  
Console.dir(a==NaN);a is not defined         
Console.dir(NaN==NaN); false  
Console.dir(8==undefined); false  
Console.dir(1==undefined); false  
Console.dir(2=={toString:function(){return2}});return2 is not defined  
Console.dir(undefined==null);true  
Console.dir(null==1);false  
Console.dir({toString:function(){return1},valueOf:function(){return []}});Object  
Console.dir(1==”1”);true  
Console.dir(1===”1”);false

145、Null,undefined的区别？

undefined表示变量声明但未初始化时的值，

null表示准备用来保存对象，还没有真正保存对象的值。从逻辑角度看，null值表示一个空对象指针。

JavaScript（ECMAScript标准）里共有5种基本类型：Undefined, Null, Boolean, Number, String，和一种复杂类型Object。可以看到null和undefined分属不同的类型，未初始化定义的值用typeof检测出来是"undefined"(字符串)，而null值用typeof检测出来是"object"（字符串）。

任何时候都不建议显式的设置一个变量为undefined，但是如果保存对象的变量还没有真正保存对象，应该设置成null。

实际上，undefined值是派生自null值的，ECMAScript标准规定对二者进行相等性测试要返回true

146、[2,6,7]和[3,5,8] => [2,3,5,6,7,8]

至少两种方法实现

 var arr1 = [2, 6, 7];  
    var arr2 = [3, 5, 8];  
    var arr=arr1.concat(arr2);  
    arr.sort(function ( a,b) {  
        return a-b;  
    });  
    console.log(arr)  
  
var arr1 = [2, 6, 7];  
    var arr2 = [3, 5, 8];  
    var arr=arr1.concat(arr2);  
    var arr3=[];  
    var min;  
    var max=Math.max.apply(null,arr)+1;  
    for(var i=0;i<arr.length;i++){  
        min=Math.min.apply(null,arr);  
        arr[arr.indexOf(min)]=max;  
        arr3.push(min)  
    }  
    console.log(arr3)

147、将一个十六进制字符串转换为十进制数

`ABC123`

148、比较一下jQuery,Angular.js,React.js 等框架的优缺点。

jQuery是继prototype之後又一个优秀的Javascrīpt框架。其宗旨是——WRITE LESS,DO MORE，写更少的代码，做更多的事情。  
它是轻量级的js库(压缩後只有21k) ，这是其它的js库所不及的，它兼容CSS3，还兼容各种浏览器 （IE 6.0+, FF 1.5+, Safari 2.0+, Opera 9.0+）。  
jQuery是一个快速的，简洁的javaScript库，使用户能更方便地处理HTML documents、events、实现动画效果，并且方便地为网站提供AJAX交互。  
jQuery还有一个比较大的优势是，它的文档说明很全，而且各种应用也说得很详细，同时还有许多成熟的插件可供选择。  
jQuery能够使用户的html页保持代码和html内容分离，也就是说，不用再在html里面插入一堆js来调用命令了，只需定义id即可。  
  
  
优点  
AngularJS是一套完整的框架，angular有自带的数据绑定、render渲染、angularUI库,过滤器,directive(模板),服务q(defer),http，inject(依赖注入),factory,provider……，等等一系列工具，基本上只要你在做web开发用过的东西，它都有一个。但是这些东西React自身都没有。  
  
Angularjs的架构清晰，分工明确，扩展性良好，model，view，controller谁在什么时候做什么事情说的很清楚，angular能够让程序员真正专注于业务逻辑，而且因为对html侵入不大，非常易于和designer协作。整个框架充满了DI的思路，耦合性非常低，对象都是被inject的，也就是说每个对象都可以轻易被替换而不影响其他对象。  
  
Angular生产效率高，单向数据流什么的想法非常好，但是写起来太麻烦！我只想变更个很简单的数据还要经过action、dispatcher、reduce、view四步，angular里一行代码就搞定的事情在react里却如此麻烦  
  
缺点  
性能   
双向数据绑定是一把双刃剑。随着组件增加，项目越来越复杂，双向数据绑定带来性能问题。   
双向数据绑定是如何影响性能的？在JavaScript（ES5）中，并没有实现当变量或对象改变时发出通知的功能，Angular的实现方法被叫做“Dirty-checking（脏检查机制）”，通过跟踪数据的改变再动态更新用户界面（UI）。   
在Angular的作用域中任何操作的执行都会引发Dirty-checking，随着绑定数量的增加性能就会越低。  
Angular 2.0推翻重做使得目前不宜采用此框架   
Angular 1.x版本其实是个比较旧的东西了，现在看来有些理念过时了，比如依赖注入、自己独特的模块化，这些东西其实在ES6下已经很好地被解决了。   
Angular的2.0几乎是一个推翻重做的框架，估计不会有1.X的upgrade方案。所以如果现在新开始的项目采用Angular的话，会是一个很尴尬的时机。同样，如此大的改动似乎也反面印证了1.X并不是那么好。  
学习成本高   
使用Angular需要学习大量的概念，包括但不限于：   
模块  
控制器  
指令  
作用域  
模板  
链式函数  
过滤器  
依赖注入  
React的优缺点  
  
优点  
  
React伟大之处就在于，提出了Virtual Dom这种新颖的思路，并且这种思路衍生出了React Native，有可能会统一Web/Native开发。在性能方面，由于运用了Virtual Dom技术，Reactjs只在调用setState的时候会更新dom，而且还是先更新Virtual Dom，然后和实际Dom比较，最后再更新实际Dom。这个过程比起Angularjs的bind方式来说，一是更新dom的次数少，二是更新dom的内容少，速度肯定快  
ReactJS更关注UI的组件化，和数据的单向更新，提出了FLUX架构的新概念，现在React可以直接用Js ES6语法了，然后通过webpack编译成浏览器兼容的ES5，开发效率上有些优势.   
React Native生成的App不是运行在WebView上，而是系统原生的UI，React通过jsx生成系统原生的UI，iOS和Android的React UI组件还是比较相似的，大量代码可以复用  
维护UI的状态,Angular 里面使用的是 $scope，在 React 里面使用的是 this.setState。 而 React 的好处在于，它更简单直观。所有的状态改变都只有唯一一个入口 this.setState()，Angular 就比较复杂，不知道背后使用了哪些黑魔法。  
同构的JavaScript   
单页面JS应用程序的最大缺陷在于对搜索引擎的索引有很大限制。React对此有了解决方案。   
React可以在服务器上预渲染应用再发送到客户端。它可以从预渲染的静态内容中恢复一样的记录到动态应用程序中。   
因为搜索引擎的爬虫程序依赖的是服务端响应而不是JavaScript的执行，预渲染你的应用有助于搜索引擎优化。  
缺点  
  
React是目标是UI组件，通常可以和其它框架组合使用，目前并不适合单独做一个完整的框架。React 即使配上 redux 的组合，也不能称之一个完整的框架，比如你想用Promise化的AJAX？对不起没有，自己找现成的库去。而且第三方组件远远不如Angular多。目前在大的稳定的项目上采用React的，我也就只知道有Yahoo的Email。React本身只是一个V而已，所以如果是大型项目想要一套完整的框架的话，也许还需要引入Redux和route相关的东西。而Angular在这方面提供的东西比React多多了.

149、请描述一下cookies，sessionStorage和localStorage的区别？

上边有

150、

var a =3;

Function test(){

var b = 2;

c = 5;

Alert(a);

}

alert(c)//结果是：

test()//结果是：

151、

var arr = [1,2,3,4,ture,’abc’,new Date()];

arr.length = 5;

alert(arr);

结果是？ 报错

152、

var arr=[1,2,3,4,5];

arr.splice(1,2,3,4,5);

alert(arr);

结果是？1,3,4,5,4,5

153、输出字符‘asdfaasfgpojgajgagjk’中出现次数最多的字符及出现的 次数。

154、合并数组[4,1,3,9,6,2]和[8,5,3,2,1,4,7]，然后去重，取出偶数倒排。

155、简述什么是面向对象。

面对对象就是：  
把数据及对数据的操作方法放在一起，作为一个相互依存的整体——对象。对同类对象抽象出其共性，形成类。类中的大多数数据，只能用本类的方法进行处理。类通过一个简单的外部接口与外界发生关系，对象与对象之间通过消息进行通信。程序流程由用户在使用中决定。

156、简述ajax过程。

1.客户端产生js的事件  
2.创建XMLHttpRequest对象  
3.对XMLHttpRequest进行配置  
4.通过AJAX引擎发送异步请求  
5.服务器端接收请求并且处理请求，返回html或者xml内容  
6.XML调用一个callback()处理响应回来的内容  
7.页面局部刷新

157、解释下时间代理。

当我们需要对很多元素添加事件的时候，可以通过将事件添加到它们的父节点而将事件委托给父节点来触发处理函数。这主要得益于浏览器的事件冒泡机制

158、解释下javascript中this是如何工作的。

函数中的this的值取决于函数调用的模式：  
方法调用模式  
当函数被保存为对象的一个属性时，成该函数为该对象的方法。函数中this的值为该对象。  
  
var foo = {  
    name: 'fooname',  
    getName: function (){  
        return C:\Users\ADMINI~1\AppData\Local\Temp\%W@GJ$ACOF(TYDYECOKVDYB.pngthis.name    
    }  
}  
foo.getName();  // this => foo  
函数调用模式  
当函数并不是对象的属性。函数中this的值为全局对象  
note：某个方法中的内部函数中的this的值也是全局对象，而非外部函数的this  
  
function foo(){  
    C:\Users\ADMINI~1\AppData\Local\Temp\%W@GJ$ACOF(TYDYECOKVDYB.pngthis.name = 'fooname';    
}  
foo();  // this => window  
构造器调用模式  
即使用new调用的函数，则其中this将会被绑定到那个新构造的对象。  
  
function Foo(){  
    C:\Users\ADMINI~1\AppData\Local\Temp\%W@GJ$ACOF(TYDYECOKVDYB.pngthis.name = 'fooname';  
}  
var foo = new Foo();  // this => foo  
使用apply或call调用模式  
该模式调用时，函数中this被绑定到apply或call方法调用时接受的第一个参数。  
  
function getName(name){  
    C:\Users\ADMINI~1\AppData\Local\Temp\%W@GJ$ACOF(TYDYECOKVDYB.pngthis.name = name;  
}  
var foo = {};  
getName.call(foo, name);  // this =>foo  
——————————————————————————  
改变this的值主要方法（目前想到的，欢迎评论添加）：  
apply或call方法调用时强制修改，使this指向第一个参数。  
使用Function.bind方法创造新的函数，该新函数的中this指向所提供的第一个参数。

159、解释下原型继承的原理。

原型继承

基本模式

代码如下:

var Parent = function(){  
    this.name = 'parent' ;  
} ;  
Parent.prototype.getName = function(){  
    return this.name ;  
} ;  
Parent.prototype.obj = {a : 1} ;

var Child = function(){  
    this.name = 'child' ;  
} ;  
Child.prototype = new Parent() ;

var parent = new Parent() ;  
var child = new Child() ;

console.log(parent.getName()) ; //parent  
console.log(child.getName()) ; //child

这种是最简单实现原型继承的方法，直接把父类的对象赋值给子类构造函数的原型，这样子类的对象就可以访问到父类以及父类构造函数的prototype中的属性

160、Call和.apply的区别是什么。

上边有

161、什么是”use strict”，使用它的好处是什么。

标记 严格模式后  
其一：如果在语法检测时发现语法问题，则整个代码块失效，并导致一个语法异常。  
其二：如果在运行期出现了违反严格模式的代码，则抛出执行异常。  
注：经过测试IE6,7,8,9均不支持严格模式。  
  
缺点：  
现在网站的[JS](https://www.baidu.com/s?wd=JS&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YknjI9PH6znjn4ujf4rH6L0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnWcsn1ndnj0LPHDzn1DLnHfvr0" \t "_blank) 都会进行压缩，一些文件用了严格模式，而另一些没有。这时这些本来是严格模式的文件，被 merge 后，这个串就到了文件的中间，不仅没有指示严格模式，反而在压缩后浪费了字节。

162、事件绑定和普通事件有什么区别

事件绑定就是针对dom元素的事件，绑定在dom元素上  
普通事件即为非针对dom元素的事件

163、请用jquery和原生js分别实现添加、移除、移动、复制、创建和查找DOM结点。

<html>   
<head>   
   <meta http-equiv="Content-Type" content="text/html; charset=utf-8">   
   <title>jQuery插入,复制、替换和删除节点</title>   
<script type="text/javascript" src="jquery-1.3.2.js"></script>   
<script type="text/javascript">   
   $(document).ready(function(){   
     //几种添加节点的方法   
    //$("p").append("<b>你好吗？</b>");//向p元素中追加《b》   
    //$("<b>你好吗？</b>").appendTo("p");//将《b》追加到p元素中   
    //$("p").prepend("<b>你好吗？</b>");//向p中前置《b》   
    //$("<b>你好吗？</b>").prependTo("p");//将《b》前置到p元素中   
    //$("p").after("<b>你好吗？</b>");//向p元素后插入《b》   
    //$("<b>你好吗？</b>").insertAfter("p");//将《b》插入到p元素后边   
    //$("p").before("<b>你好吗？</b>");//在p元素之前添加《b》   
    //$("<b>你好吗？</b>").insertBefore("p");//将《b》插入到p元素前面   
  
     
      //几种删除节点的方法   
    //var $li=$("ul li:eq(1)").remove();//删除ul节点中第2个元素节点   
    //$("ul").append($li);//把刚删除的元素节点从新添加到ul元素中去   
    //$("ul li").remove("li[title!=菠萝]");//将ul元素下title属性不等于"菠萝"的li元素删除   
    //$("ul li:eq(1)").empty();//清空ul节点下第2个li元素的内容   
  
    //复制节点   
   /\* $("ul li").click(function(){   
     $(this).clone(true).appendTo("ul");//复制当前点击的节点，并将它追加到《ul》元素中，当添加参数时复制它的事件   
  
    });   
    \*/   
    //替换节点   
   // $("p").replaceWith("<strong>你最不喜欢的水果是？</Strong>");     
  
   });   
</script>   
</head>   
<body>   
   <p>你好！</p>   
   你最喜欢的水果是？   
   <ul>   
    <li title="苹果">苹果</li>   
    <li title="橘子">橘子</li>   
    <li title="菠萝">菠萝</li>   
    </ul>   
</body>   
</html>

164、实时监测用户在input内输入的字符数应该监听哪个事件。

首先想到的是change事件，但用过change的都知道只有在input失去焦点时才会触发，并不能满足实时监测的需求，比如监测用户输入字符数。  
在经过查阅一番资料后，欣慰的发现firefox等现代浏览器的input有oninput这一属性，可以用三种方式使用它：  
1,内嵌元素方式(属性编辑方式)  
<input id="test" oninput="console.log('input');" type="text" />  
2,句柄编辑方式  
document.getElementById('test').oninput=function(){  
    console.log('input');  
}  
3,事件侦听方式(jquery)  
$('#test').on('input',function(){ console.log('input'); })  
 但是，以上代码仅在除了ie的浏览器大大们里才work，那ie该怎么处理呢？ 在ie中有一个属性叫做onpropertychange：  
<input id="test" onpropertychange="alert('change');" type="text" />  
经过调试后马上就会发现，这个属性是在元素的任何属性变化时都会起作用，包括我们这里所提到的value，但至少是起作用了，那接下来的任务就是筛选出property为value的变化。  
  
document.getElementById('test').attachEvent('onpropertychange',function(e) { if(e.propertyName!='value') return;  
    $(that).trigger('input');  
});  
在上面代码中的回调函数中会传入一个参数，为该事件，该事件有很多属性值，搜寻一下可以发现有一个我们很关心的，叫做propertyName，也就是当前发生变化的属性名称。然后就相当简单了，只要在回调函数中判断一下是否为我们所要的value，是的话就trigger一下‘input’事件。  
  
然后，就可以在主流浏览器中统一用这样的方式来监听‘input’事件了。  
  
$('#test').on('input',function(){  
    alert('input');  
})  
  
完整代码：  
$('#test').on('input',function(){  
    alert('input');  
}) //for ie if(document.all){  
    $('input[type="text"]').each(function() { var that=this; if(this.attachEvent) { this.attachEvent('onpropertychange',function(e) { if(e.propertyName!='value') return;  
                $(that).trigger('input');  
            });  
        }  
    })  
}

165、介绍下javascript原型，原型链的特点。

每个对象都会在其内部初始化一个属性，就是prototype(原型)，当我们访问一个对象的属性时，

如果这个对象内部不存在这个属性，那么他就会去prototype里找这个属性，这个prototype又会有自己的prototype，

于是就这样一直找下去，也就是我们平时所说的原型链的概念。

特点：

JavaScript对象是通过引用来传递的，我们创建的每个新对象实体中并没有一份属于自己的原型副本。当我们修改原型时，与之相关的对象也会继承这一改变。

167、判断一个字符串中出现次数最多的字符，统计这个次数。

168、JS判断数据类型的方法。

1、最常见的判断方法：typeof  
  
alert(typeof a)   ------------> string  
alert(typeof b)   ------------> number  
alert(typeof c)   ------------> object  
alert(typeof d)   ------------> object  
alert(typeof e)   ------------> function  
alert(typeof f)   ------------> function  
其中typeof返回的类型都是字符串形式，需注意，例如：  
alert(typeof a == "string") -------------> true  
alert(typeof a == String) ---------------> false  
另外typeof 可以判断function的类型；在判断除Object类型的对象时比较方便。  
2、判断已知对象类型的方法： instanceof  
  
alert(c instanceof Array) ---------------> true  
alert(d instanceof Date)   
alert(f instanceof Function) ------------> true  
alert(f instanceof function) ------------> false  
注意：instanceof 后面一定要是对象类型，并且大小写不能错，该方法适合一些条件选择或分支。  
3、根据对象的constructor判断： constructor  
  
alert(c.constructor === Array) ----------> true  
alert(d.constructor === Date) -----------> true  
alert(e.constructor === Function) -------> true  
注意： constructor 在类继承时会出错  
eg：  
      function A(){};  
      function B(){};  
      A.prototype = new B(); //A继承自B  
      var aObj = new A();  
      alert(aobj.constructor === B) -----------> true;  
      alert(aobj.constructor === A) -----------> false;  
而instanceof方法不会出现该问题，对象直接继承和间接继承的都会报true：  
      alert(aobj instanceof B) ----------------> true;  
      alert(aobj instanceof B) ----------------> true;  
言归正传，解决construtor的问题通常是让对象的constructor手动指向自己：  
      aobj.constructor = A; //将自己的类赋值给对象的constructor属性  
      alert(aobj.constructor === A) -----------> true;  
      alert(aobj.constructor === B) -----------> false; //基类不会报true了;  
4、通用但很繁琐的方法： prototype  
  
alert(Object.prototype.toString.call(a) === ‘[object String]’) -------> true;  
alert(Object.prototype.toString.call(b) === ‘[object Number]’) -------> true;  
alert(Object.prototype.toString.call(c) === ‘[object Array]’) -------> true;  
alert(Object.prototype.toString.call(d) === ‘[object Date]’) -------> true;  
alert(Object.prototype.toString.call(e) === ‘[object Function]’) -------> true;  
alert(Object.prototype.toString.call(f) === ‘[object Function]’) -------> true;  
大小写不能写错，比较麻烦，但胜在通用。  
5、无敌万能的方法：jquery.type()  
  
如果对象是undefined或null，则返回相应的“undefined”或“null”。  
jQuery.type( undefined ) === "undefined"  
jQuery.type() === "undefined"  
jQuery.type( window.notDefined ) === "undefined"  
jQuery.type( null ) === "null"  
如果对象有一个内部的[[Class]]和一个浏览器的内置对象的 [[Class]] 相同，我们返回相应的 [[Class]] 名字。 (有关此技术的更多细节。 )  
jQuery.type( true ) === "boolean"  
jQuery.type( 3 ) === "number"  
jQuery.type( "test" ) === "string"  
jQuery.type( function(){} ) === "function"  
jQuery.type( [] ) === "array"  
jQuery.type( new Date() ) === "date"  
jQuery.type( new Error() ) === "error" // as of jQuery 1.9  
jQuery.type( /test/ ) === "regexp"  
其他一切都将返回它的类型“object”。

169、jquery怎样获取div的id为‘content’的文本内容。

$(“#content”).html()

170、如何阻止事件冒泡和默认事件，用js原生代码说明。

1. event.preventDefault();  -- 阻止元素的默认事件。  
注：a元素的点击跳转的默认事件 ，  
button，radio等表单元素的默认事件 ，  
div 元素没有默认事件  
例：  
代码如下:  
  
<a href="C:\Users\ADMINI~1\AppData\Local\Temp\[5UQ[BL(6~BS2JV6W}N6[%S.pnghttp://www.baidu.com" target="\_black">百度</a>  
代码如下:  
  
var samp = document.getElementByTagName("a");  
samp.addEventListener("click",function(e){e.preventDefault()},false);  
解释：点击链接的时候正常情况下会发生跳转，但是现在我们阻止了它的默认事件，即跳转事件，这时就不会跳转到百度了。  
  
2. event.stopPropagation();  -- 阻止元素冒泡事件  
注：嵌套元素一般都存在冒泡事件，会带来某些影响  
例：  
代码如下:  
  
<div id="c1" onclick="alert(1)">  
<div id="c2" onlick="alert(2)">  
<input type="button" id="c3" value="点击" onclick="alert(3)">  
</div>  
</div>  
这里点击button的时候，浏览器会先后弹出3,2,1，本来只想让绑定在button上的事件发生,却无意中触发了它的两个父级上的事件，这里我们只是做了一个简单测试，试想如果在项目开发中，某个按钮和他的父级同时绑定了很重要的事件，那么结果会惨不忍睹。这时的处理方法就是阻止冒泡事件。  
给input注册click事件，同时阻止它的冒泡事件  
代码如下:  
  
document.getElementById('c3').addEventListener('click',function(e){e.stopPropagation()},false);

171、js中如何对特殊符号进行转码与编码。

js对文字进行编码涉及3个函数：escape,encodeURI,encodeURIComponent，相应3个解码函数：unescape,decodeURI,decodeURIComponent  
1、   传递参数时需要使用encodeURIComponent，这样组合的url才不会被#等特殊字符截断。                              
例如：<script language="javascript">document.write('<a href="http://passport.baidu.com/?logout&aid=7&u= +encodeURIComponent("http://cang.baidu.com/bruce42")+">退出</a>');</script>  
2、   进行url跳转时可以整体使用encodeURI  
例如：Location.href=encodeURI("http://cang.baidu.com/do/s?word=百度&ct=21");  
3、   js使用数据时可以使用escape  
例如：搜藏中history纪录。  
4、   escape对0-255以外的unicode值进行编码时输出%u\*\*\*\*格式，其它情况下escape，encodeURI，encodeURIComponent编码结果相同。  
最多使用的应为encodeURIComponent，它是将中文、韩文等特殊字符转换成utf-8格式的url编码，所以如果给后台传递参数需要使用encodeURIComponent时需要后台解码对utf-8支持（form中的编码方式和当前页面编码方式相同）  
escape不编码字符有69个：\*，+，-，.，/，@，\_，0-9，a-z，A-Z  
encodeURI不编码字符有82个：!，#，$，&，'，(，)，\*，+，,，-，.，/，:，;，=，?，@，\_，~，0-9，a-z，A-Z  
encodeURIComponent不编码字符有71个：!， '，(，)，\*，-，.，\_，~，0-9，a-z，A-Z

172、谈谈你对跨域访问的认识，写出两个实现跨域访问页面的的方法。

上边有

173、列出你所知道的设计模式并简述。

**一 单例模式**

单例模式的定义是产生一个类的唯一实例，但js本身是一种“无类”语言。很多讲js设计模式的文章把{}当成一个单例来使用也勉强说得通。因为js生成对象的方式有很多种，我们来看下另一种更有意义的单例。

有这样一个常见的需求，点击某个按钮的时候需要在页面弹出一个遮罩层。比如web.qq.com点击登录的时候.

**二 简单工厂模式**

简单工厂模式是由一个方法来决定到底要创建哪个类的实例, 而这些实例经常都拥有相同的接口. 这种模式主要用在所实例化的类型在编译期并不能确定， 而是在执行期决定的情况。 说的通俗点，就像公司茶水间的饮料机，要咖啡还是牛奶取决于你按哪个按钮。

简单工厂模式在创建ajax对象的时候也非常有用.

**三 观察者模式**

观察者模式( 又叫发布者-订阅者模式 )应该是最常用的模式之一. 在很多语言里都得到大量应用. 包括我们平时接触的dom事件. 也是js和dom之间实现的一种观察者模式.

**四 适配器模式**

去年年前当时正在开发dev.qplus.com, 有个存储应用分类id的js文件, 分类id的结构最开始设计的比较笨重. 于是我决定重构它. 我把它定义成一个json树的形式

174、javascr中有哪些数据类型；

上边有

176、什么是闭包，请举例说明；

上边有

177、一个dom节点被点击的时候，我们希望能够执行一个函数，应该怎么做；

178、请写一个javascript函数parseQueryString，他的用途是吧url参数解析为一个对象， 如：var url = <http://www.test.com/index.php?key0=0&key1=1&key2=2>

上边有

179、怎样添加移除、移动、复制、创建和查找节点？

上边有！！！

180、在javascript中什么是伪数组，如何将伪数组转化为标准数组；

上边有

181、手写数组快速排序；

182、统计字符号串“aaaabbbccccddfggh”中字母个数或统计最多字母数；

上边有

183、写一个function，清楚字符串前后的空格（兼容所有浏览器）

184、实现一个函数clone，可以对javascript中5种主要的数据类型（包括number string object array boolean）进行值复制；

上边有

185、将数组【“a”，“b”，“c”】转化为字符串；

<**script**>  
 var ***a***, ***b***;  
 ***a*** = new Array(a,b,c);  
 ***b*** = ***a***.join("-");  
</**script**>

186、获取enent目标对象的方法（要求兼容性）；

IE: 可以直接用window.event或event获得event对象,同时可直接用event.x或event.y获得当前操作位置的x和y坐标,当然还有其它属性,如:srcElement,keyCode等  
Firefox:不能直接使用window.event和event对象,当然也就不能获得该对象的其它属性.Firefox下获得操作位置坐标的属性是:pageX和pageY;   
下面提供同时兼容IE和Firefox的获得event对象的方法,在需要用到event对象的地方,调用该方法即可.  
function getEvent() //同时兼容ie和ff的写法   
    {    
        if(document.all)  return window.event;      
        func=getEvent.caller;          
        while(func!=null){    
            var arg0=func.arguments[0];   
            if(arg0)   
            {   
              if((arg0.constructor==Event || arg0.constructor ==MouseEvent) || (typeof(arg0)=="object" && arg0.preventDefault && arg0.stopPropagation))  
              {    
              return arg0;   
              }   
            }   
            func=func.caller;   
        }   
        return null;   
    }   
  
如调用:function test() {   
        var event=getEvent();   
        if (navigator.appName=='Microsoft Internet Explorer') {   
            alert(event.x);   
        }else if (navigator.appName=='Netscape') {   
            alert(event.pageX);   
        }   
          
    }

187、介绍js基本数据类型；

上边有

188、null undefined 的区别；

上边有

189、javascript中“use strict”是什么意思，为什么要使用；

上边有

190、描述一下javascript 的作用域链；

任何一段JavaScript代码都对应一个作用域链，作用域链中存放一系列对象，代码中声明的变量将作为对象的属性存放。  
在JavaScript的顶层代码中，作用域链由一个全局对象组成；当定义一个函数时，它保存一个作用域链，作用域链上有两个对象，一个是函数对象，一个是全局对象。  
每当一个函数被调用时，会创建一个活动对象（也叫上下文对象），函数中的局部变量将作为该对象的属性存放。  
当需要使用一个变量时，将从作用域链中逐个查找对象的属性。比如：要使用变量a，将先查找作用域中的第一个对象是否有属性a，如果有就使用；如果没有就查找作用域链中下一个对象的属性，以此类推。如果作用域链上没有任何一个对象含有属性 x，则认为这段代码的作用域链上不存在x，将抛出引用错误异常。  
当函数调用完成后，如果没有其他引用指向为此次调用所创建的上下文对象，该对象将被回收。  
var a = 100; function f() { var a = 200; function g() { return a; } return g; } console.log(f()());  
上述代码运行，会输出200。   
上述代码意味着，先执行 f()，得到该函数内嵌套的一个函数对象 g，然后调用这个嵌套的函数。   
定义函数 f 时，它保存一个作用域链，作用域链上有两个对象，一个是函数对象，一个是全局对象，此时，全局变量 a 是全局对象的属性，和函数对象 f 无关。   
执行函数 f 时，会创建一个活动对象，其中保存变量 a 作为函数 f 的属性而存在。而函数g嵌套定义在函数 f里，对其访问变量 a时，它没有变量 a，则继续往下查找函数 f。找到函数 f 的属性 a 并返回。因此，输出结果为 200。   
这种函数将变量包裹起来，外部代码可以通过内嵌函数 g来访问函数 f 内的局部变量的方式，也称为闭包。

191、写一个正则，验证电话号码：格式为：“区号-号码”其中区号是010或者20N、0NNN， 电话号码是7-8位数字。

192、编写一个方法，出去数组中的重复的元素：例如数字【“ab”，“ab”，“ac”，“cd”】 处 理后、：【“ab”，“ac”，“cd”】

193、如何隐藏一个dom元素

设置css display属性为none,效果：元素不显示，不占位  
设置css visibility属性为hidden,效果：元素不显示，但占位

194、document.write、innerHTML和innerText 的区别？

document.write是直接写入到页面的内容流，如果在写之前没有调用document.open, 浏览器会自动调用open。每次写完关闭之后重新调用该函数，会导致页面被重写。 innerHTML则是DOM页面元素的一个属性，代表该元素的html内容。你可以精确到某一个具体的元素来进行更改。如果想修改document的内容，则需要修改document.documentElement.innerElement。 innerHTML将内容写入某个DOM节点，不会导致页面全部重绘 innerHTML很多情况下都优于document.write，其原因在于其允许更精确的控制要刷新页面的那一个部分。

195、javascript 中有哪些基础数据类型

上边有

196、如何添加HTML事件，有几种方法？（至少两种方式）

1. 通过HTML元素属性。简单说来就是在html结构中，给你要添加事件的元素添加一个属性。  
属性名为 'on' + 事件名。  
如:你要给a元素绑定一个click事件，你就该这么写：  
<a href="C:\Users\ADMINI~1\AppData\Local\Temp\%W@GJ$ACOF(TYDYECOKVDYB.pnghttp://blog.163.com/luping\_01/blog/#" onclick='do something'>name</a>  
       2.通过对象属性。  
对象指的是jDOM树里的对象，我们都知道，所有的html元素在DOM(文档对象类型)里都存在一个相应的DOM元素。  
给这个DOM元素添加事件等同于方式一。这个属性名也一样：  
也为 on + 事件名  
如：  
html结构:  
<a href="C:\Users\ADMINI~1\AppData\Local\Temp\%W@GJ$ACOF(TYDYECOKVDYB.pnghttp://blog.163.com/luping\_01/blog/#" id="n">name</a>  
js：  
document.getElementById('n').onclick = function(){ //do something}  
     3.通过W3C监听方式（标准方式）或者IE专属的中间模型添加事件  
W3C方式： elemennt.addEventListener(事件名,处理函数引用,true || false)  
这里的事件名就是单纯的事件名，不需要加 on ，true表示在事件传播阶段捕获事件，false表示在事件冒泡阶段捕获事件，一般我们设为false!  
IE模式: element.attachEvent( 'on' + 事件名，处理函数引用);  
注意，IE模式需要加on

197、javascript 面向对象中继承实现方式

**一、原型链继承**

**二、使用apply、call方法**

**三、对象实例间的继承**

198、javascript 中如何让对一个对象进行深拷贝？

199、为什么异步加载js文件？加载方式？（至少两种方式）

200、用js或jQuery获取id为chuchujie的div里边的所有a标签，并把字体大小设置为

18px；同 时把第三个a标签设置为块级元素，并把他的宽高分别设置为300px和500px；

 var as = document.getElementById('chuanchuanjie').getElementsByTagName('a');  
        for(var i = 0,len = as.length;i<len;i++){  
            if(i==2){  
                as[i].style.display = 'block';  
                as[i].style.height = '500px';  
                as[i].style.width = '300px';  
            }  
            as[i].style.fontSize = '18px';  
        }

201、请在网页上输出九九乘法表；

<**script**>  
 for (var ***i*** = 1; ***i*** < 10; ***i***++) {  
 for (var ***j*** = ***i***; ***j*** > 0; ***j***--) {  
 **document**.write(***i*** + "\*" + ***j*** + "=" + ***i*** \* ***j***);  
 **document**.write(" ")  
 }  
 **document**.write("<br/>")  
 }  
</**script**>

202、call 和apply的区别是什么？

203、请解释jsonp的工作原理

可以直接理解为服务端输出[JS代码](https://www.baidu.com/s?wd=JS%E4%BB%A3%E7%A0%81&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YkuWfduWfvrjPhuhf3njR30ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3En10vrjT1P1nvn1DvPHmdPHmY" \t "_blank)：  
客户端POST或GET请求都没关系重点在服务端输出的结果。  
首先响应结果中Content-Type=text/javascript  
然后前端代码直接以

204、请简要说明javascript 中this 是如何让工作的的

JavaScript 只有函数能提供作用域，this 表达了这个作用域的上下文。这个上下文是一个对象，默认是 global 对象：  
  
function foo() {  
  
  // 在终端输出 this 对象  
  console.log(this)  
}  
  
foo() // 输出： global 对象 在浏览器中即 window 对象  
  
但是也可以在函数被运行的时候动态指定（call,apply, bind）：

205、请说出至少三种减少页面加载我时间的方法。（加载时间是指感知的时间或者实际加 载的时间）

1. 重复的HTTP请求数量应尽量减少

2. 服务器启用gzip压缩功能

3. 在文件头部放置css样式的定义

4. 文件末尾放Javascript脚本

5. 压缩Javascript、CSS代码

6. Ajax采用缓存调

7. 尽可能减少DCOM元素

206、请描述你曾使用过的css预处理器的优缺点

缺点：简单来说CSS预处理器语言较CSS玩法变得更高级了，但同时降低了自己对最终代码的控制力。更致命的是提高了门槛，首先是上手门槛，其次是维护门槛，再来是团队整体水平和规范的门槛。这也造成了初学学习成本的昂贵。  
优点：用一种专门的编程语言，为CSS增加了一些编程的特性，将CSS作为目标生成文件，然后开发者就只要使用这种语言进行编码工作。通俗的说，CSS预处理器用一种专门的编程语言，进行Web页面样式设计，然后再编译成正常的CSS文件，以供项目使用。CSS预处理器为CSS增加一些编程的特性，无需考虑浏览器的兼容性问题，例如你可以在CSS中使用变量、简单的逻辑程序、函数等等在编程语言中的一些基本特性，可以让你的CSS更加简洁、适应性更强、可读性更佳，更易于代码的维护等诸多好处。

207、请谈谈你都使用过哪些javascript模板

上边有

208、你怎么看AMD vs.Commonjs?

js开发者对js模块加载的尝试和创新从来都没有停止过，尤其是当nodejs的出现后，模块化加载的必要性更加凸显。本文不讨论如何在nodejs环境来模块化加载（创造者已经利用commonJS机制解决），只讨论在浏览器环境下如何来模块加载的思路，并提出一些我的看法。  
  
浏览器环境与nodejs的环境的最大差异是，对于nodejs的环境，大多数情况下被依赖的模块文件本身就在本地（它们都在服务器上），同步取过来就能用；而对于浏览器的环境，被依赖的模块文件通常还在远程服务器上，并未加载到本地，也就是说必须是先加载（并解析）后执行的机制。  
  
既然已经有了commonJS，在这之上将异步回调的逻辑加入进去，可是异步先加载什么呢？于是就有了依赖的概念。一段时间的发展后，有了AMD、和CMD的解决方案，代表作品是requirejs和seajs，有兴趣的读者可以去了解一下，这里就不展开介绍了。  
有必要简单提一下两者的主要区别，CMD推崇依赖就近，可以把依赖写进你的代码中的任意一行，例：  
define(function(require, exports, module) {  
  var a = require('./a')  
  a.doSomething()  
  var b = require('./b')  
  b.doSomething()  
})  
代码在运行时，首先是不知道依赖的，需要遍历所有的require关键字，找出后面的依赖。具体做法是将function toString后，用正则匹配出require关键字后面的依赖。显然，这是一种牺牲性能来换取更多开发便利的方法。  
  
而AMD是依赖前置的，换句话说，在解析和执行当前模块之前，模块作者必须指明当前模块所依赖的模块，表现在require函数的调用结构上为：  
define(['./a','./b'],function(a,b){  
   a.doSomething()  
   b.doSomething()  
})  
代码在一旦运行到此处，能立即知晓依赖。而无需遍历整个函数体找到它的依赖，因此性能有所提升，缺点就是开发者必须显式得指明依赖——这会使得开发工作量变大，比如：当你写到函数体内部几百上千行的时候，忽然发现需要增加一个依赖，你不得不回到函数顶端来将这个依赖添加进数组。  
  
细心的读者可能发现，到目前位置我讨论的AMD和CMD的思想的关于依赖的部分，都只讨论的“硬依赖”，也就是执行前肯定需要的依赖，但是这不是全部的情况。有的时候情况是这样的：  
// 函数体内：  
if(status){  
  a.doSomething()  
}  
在这个函数体内，可能依赖a，也可能不依赖a，我把这种可能的依赖成为“软依赖”。对于软依赖当然可以直接当硬依赖处理，但是这样不经济，因为依赖是不一定的，有可能加载了此处的依赖而实际上没有用上。  
对于软依赖的处理，我推荐依赖前置+回调函数的实现形式。上面的例子简单表述如下：  
// 函数体内：  
if(status){  
  async(['a'],function(a){  
    a.doSomething()  
  })  
}  
至此可以对由commonJS衍生出来的方案做出总结了。在浏览器端来设计模块加载机制，需要考虑依赖的问题。  
我们先把依赖分为两种，“强依赖” —— 肯定需要 和“弱依赖” —— 可能需要。  
对于强依赖，如果要性能优先，则考虑参照依赖前置的思想设计你的模块加载器，我个人也更推崇这个方案一些；如果考虑开发成本优先，则考虑按照依赖就近的思想设计你的模块加载器。  
对于弱依赖，只需要将弱依赖的部分改写到回调函数内即可。  
如果现在我要实现一个模块加载器，我会将强依赖前置，弱依赖采用异步回调函数的形式，其它的方法我认为都只是语法糖而已，仅此就够了。

**第三阶段**

Html

什么是Web workers？为什么我们需要他

Web Workers为WEB前端网页上的脚本提供了一种能在后台进程中运行的方法。一旦它被创建，Web Workers就可以通过postMessage()向任务池发送任务请求，执行完之后再通过postMessage()返回消息给创建者指定的事件处理程序(通过onmessage进行捕获)。   
  
Web Workers进程能够在不影响用户界面的情况下处理任务，并且，它还可以使用XMLHttpRequest来处理I/O，无论responseXML和channel属性是否为null。

列一个手机端可以实现手势滑动的插件或类库

请说一下移动端常见的适配不同屏幕大小的方法。

<meta name="viewport" content="width=device-width, initial-scale=1"/>

然后网页宽度一般都用百分比，不要固定，再用@media媒体查询，不同大小屏幕要做不同的css样式，附件是一个@media的小例子，缩放浏览器窗口可以查看样式的变化

什么是css预处理器

本地存储和cookies'之间的区别是什么

HTML5本地存储是一种让网页可以把键值对存储在用户浏览器客户端的方法。像Cookie一样，这些数据不会因为你打开新网站，刷新页面，乃至关闭你的浏览器而消失。 而与Cookie不同的时，这些数据不会每次随着[HTTP请求](https://www.baidu.com/s?wd=HTTP%E8%AF%B7%E6%B1%82&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YkuywhuAnzm1FBuHTzPWfk0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPHR1n1TLrHDknHRvP16YrjDz" \t "_blank)被发送到[服务器端](https://www.baidu.com/s?wd=%E6%9C%8D%E5%8A%A1%E5%99%A8%E7%AB%AF&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YkuywhuAnzm1FBuHTzPWfk0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPHR1n1TLrHDknHRvP16YrjDz" \t "_blank)(当然如果你需要这么做，你可以自己编程实现 ）. 因为这是HTML5规范的一部分，这一接口会被浏览器原生支持，不用依赖任何第三方插件。

get和post区别？

Js

1、jQuery this与this区别

2、input name a与p name a 这两个如何用jQuery定位到input

$("input[name='a']")

3、请描述一下 cookies，sessionStorage 和 localStorage 的区别？

* **sessionStorage 、localStorage 和 cookie 之间的区别**  
  共同点：都是保存在浏览器端，且同源的。
* 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；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。
* 而sessionStorage和localStorage不会自动把数据发给服务器，仅在本地保存。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。

4、ajax 是什么

Ajax是几个简单首字母的所写：Asynchronous JavaScript and XML （异步的JavaScript和Xml），Ajax并不是全新的技术，而是整合了几种现有的技术：JavaScript、XML与CSS。主要是JavaScript，我们通过 JavaScript的XMLHttpRequest对象完成发送请求到服务器并返回结果的任务，然后使用JavaScript更新局部的网页。异步指的是，JavaScript脚本发送请求后并不是一直等着服务器响应，而是发送请求后继续做别的事，请求响应和处理是异步完成的。XML一般用于请求数据和响应数据的封装，CSS用于美化页面样式

5、同步和异步的区别?如何解决跨域问题

1. 同步交互：指发送一个请求,需要等待返回,然后才能够发送下一个请求，有个等待过程；
2. 异步交互：指发送一个请求,不需要等待返回,随时可以再发送下一个请求，即不需要等待。 区别：一个需要等待，一个不需要等待，在部分情况下，我们的项目开发中都会优先选择不需要等待的异步交互方式。

6、异步加载的方式有哪些

1.defer:defer  
　　JS异步下载，dom结构解析完（标签 + 样式（内容不一定下载完））才异步执行   
　　仅IE能用  
　　内部JS也能用该属性  
　　异步加载js不允许使用document.write，因为document.write会清除文档流，js标签还未加载就会被清除  
　　document.write()可用于初始化页面  
2.(h5)async:async（asynchronous） ajax(asynchronous javascript and XML)  
　　JS异步加载，加载完毕后立刻异步执行  
　　IE8及以下不兼容  
　　内部JS不能用该属性  
7、WEB应用从服务器主动推送Data到客户端有那些方式

主要就是post和get两种传递方式

8、juery中如何将数组转化为json字符串，然后在转化回来

$.fn.stringify = function() {  
  return JSON.stringify(this);  
}  
然后这样调用：  
  
1  
$(array).stringify();

9、给juery扩展一个新的空方法，方法名称为newFunc

10、ajax请求时，如何解释json数据

jquery有自带的eval()方法，用法如下:  
$.ajax({  
...  
success: function(data) {  
var obj = eval("("+data+")"); // 解析  
}  
});  
  
上面的obj就是解析后的json对象了，当然你要保证ajax请求成功有返回值

11、jquery 中哪几种类型选择器？

层叠选择器：

基本过滤选择器

内容过滤选择器

可视化过滤选择器

属性过滤选择器

子元素过滤选择器

表单元素选择器

表单元素过滤选择器

12、sessionStorage、localStorage、cookie的区别

sessionStorage 和 localStorage 是HTML5 Web Storage API 提供的，可以方便的在web请求之间保存数据。有了本地数据，就可以避免数据在浏览器和服务器间不必要地来回传递。  
sessionStorage、localStorage、cookie都是在浏览器端存储的数据，其中sessionStorage的概念很特别，引入了一个“浏览器窗口”的概念。sessionStorage是在同源的同窗口（或tab）中，始终存在的数据。也就是说只要这个浏览器窗口没有关闭，即使刷新页面或进入同源另一页面，数据仍然存在。关闭窗口后，sessionStorage即被销毁。同时“独立”打开的不同窗口，即使是同一页面，sessionStorage对象也是不同的。

13、ajax执行步骤

第一步：客户端触发异步操作

第二步：创建新的XMLHttpRequest对象(核心部分)

第三步：告诉XMLHttpRequest对象哪个函数会处理XMLHttpRequest对象状态的改变，为此要把对象的onReadyStateChange属性设置为响应该事件的JavaScript函数的引用

第四步：指定请求的属性，包括提交方式(get OR post)，url，true or false，XMLHttpRequest对象的open()方法会指定将发出的请求。

第五步：将请求发送给服务器端。XMLHttpRequest对象的send()方法把请求发送到指定的目标资源。

第六步：XMLHttpRequest对象接收处理结果并分析

第七步：更新页面 一共需要三个方法： 第一个方法：创建XMLHttpRequest对象的方法(固定不变的) 第二种方法：实现Ajax的主方法，包含了一步一步的过程，需要调用第一个方法，并将第三个方法指定给XMLHttpRequest对象的onreadystatechange事件 第三个方法：动态的改变我们页面的方法，通过这个方法可以动态的改变我们的页面

14、XTMHTTPRequest的readystate的五种状态？

15、status返回的窗口状态栏文本类型？（或http的状态码？）

16、写一个ajax使用get请求数据的全套代码？

<script type="text/javascript">  
    /\* 创建 XMLHttpRequest 对象 \*/  
    var xmlHttp;  
    function GetXmlHttpObject(){  
        if (window.XMLHttpRequest){  
            // code for IE7+, Firefox, Chrome, Opera, Safari   
            xmlhttp=new XMLHttpRequest();  
        }else{// code for IE6, IE5   
            xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");  
        }  
        return xmlhttp;  
    }  
    // -----------ajax方法-----------//   
    function getLabelsGet(){  
        xmlHttp=GetXmlHttpObject();  
        if (xmlHttp==null){  
            alert('您的浏览器不支持AJAX！');  
            return;  
        }  
        var id = document.getElementById('id').value;  
        var url="C:\Users\ADMINI~1\AppData\Local\Temp\%W@GJ$ACOF(TYDYECOKVDYB.pnghttp://www.Leefrom.com?id="+id+"&t/"+Math.random();  
        xmlHttp.open("GET",url,true);  
        xmlHttp.onreadystatechange=favorOK;//发送事件后，收到信息了调用函数   
        xmlHttp.send();  
    }  
    function getOkGet(){  
        if(xmlHttp.readyState==1||xmlHttp.readyState==2||xmlHttp.readyState==3){  
            // 本地提示：加载中   
        }  
        if (xmlHttp.readyState==4 && xmlHttp.status==200){  
            var d= xmlHttp.responseText;  
            // 处理返回结果   
        }  
    }  
</script>

17、ajax跨域解决方案？

$(document).ready(function(){

var url='http://localhost:8080/WorkGroupManagment/open/getGroupById"

+"?id=1&callback=?';

$.ajax({

url:url,

dataType:'jsonp',

processData: false,

type:'get',

success:function(data){

alert(data.name);

},

error:function(XMLHttpRequest, textStatus, errorThrown) {

alert(XMLHttpRequest.status);

alert(XMLHttpRequest.readyState);

alert(textStatus);

}});

});

18、在什么情况下你会用ajax请求来代替通常的跳转

局部刷新

19、谈谈你对跨域访问的认识，写出两个实现跨域访问页面的的方法

以上有

20、列出你所知道的设计模式并简述

1、单例模式：控制对象个数  
2、装饰者模式：Java的IO系统中使用了较多的装饰模式，主要用于给一个对象添加更多更好的功能，比如说给输入输出流添加缓冲  
3、工厂模式：需要使用对象的时候使用工厂来创建对象  
4、抽象工厂模式：可以创建多个系统的对象  
5、适配器模式：在不同的系统之间做耦合的作用  
6、建造者模式：将创建过程封装起来  
7、观察者模式：当相应的事件发生的时候，通知注册了的对象，比如说ide的双击事件等

21、Ajax 交换模型？同步和异步区别

以上有

22、用js或jQuery获取id为chuchujie的div里边的所有a标签，并把字体大小设置为18px； 同时把第三个a标签设置为块级元素，并把他的宽高分别设置为300px和500px；

23、请解释jsonp的工作原理

24、你怎么看AMD vs.Commonjs?

js开发者对js模块加载的尝试和创新从来都没有停止过，尤其是当nodejs的出现后，模块化加载的必要性更加凸显。本文不讨论如何在nodejs环境来模块化加载（创造者已经利用commonJS机制解决），只讨论在浏览器环境下如何来模块加载的思路，并提出一些我的看法。  
  
浏览器环境与nodejs的环境的最大差异是，对于nodejs的环境，大多数情况下被依赖的模块文件本身就在本地（它们都在服务器上），同步取过来就能用；而对于浏览器的环境，被依赖的模块文件通常还在远程服务器上，并未加载到本地，也就是说必须是先加载（并解析）后执行的机制。  
  
既然已经有了commonJS，在这之上将异步回调的逻辑加入进去，可是异步先加载什么呢？于是就有了依赖的概念。一段时间的发展后，有了AMD、和CMD的解决方案，代表作品是requirejs和seajs，有兴趣的读者可以去了解一下，这里就不展开介绍了。  
有必要简单提一下两者的主要区别，CMD推崇依赖就近，可以把依赖写进你的代码中的任意一行，例：  
define(function(require, exports, module) {  
  var a = require('./a')  
  a.doSomething()  
  var b = require('./b')  
  b.doSomething()  
})  
代码在运行时，首先是不知道依赖的，需要遍历所有的require关键字，找出后面的依赖。具体做法是将function toString后，用正则匹配出require关键字后面的依赖。显然，这是一种牺牲性能来换取更多开发便利的方法。  
  
而AMD是依赖前置的，换句话说，在解析和执行当前模块之前，模块作者必须指明当前模块所依赖的模块，表现在require函数的调用结构上为：  
define(['./a','./b'],function(a,b){  
   a.doSomething()  
   b.doSomething()  
})  
代码在一旦运行到此处，能立即知晓依赖。而无需遍历整个函数体找到它的依赖，因此性能有所提升，缺点就是开发者必须显式得指明依赖——这会使得开发工作量变大，比如：当你写到函数体内部几百上千行的时候，忽然发现需要增加一个依赖，你不得不回到函数顶端来将这个依赖添加进数组。  
  
细心的读者可能发现，到目前位置我讨论的AMD和CMD的思想的关于依赖的部分，都只讨论的“硬依赖”，也就是执行前肯定需要的依赖，但是这不是全部的情况。有的时候情况是这样的：  
// 函数体内：  
if(status){  
  a.doSomething()  
}  
在这个函数体内，可能依赖a，也可能不依赖a，我把这种可能的依赖成为“软依赖”。对于软依赖当然可以直接当硬依赖处理，但是这样不经济，因为依赖是不一定的，有可能加载了此处的依赖而实际上没有用上。  
对于软依赖的处理，我推荐依赖前置+回调函数的实现形式。上面的例子简单表述如下：  
// 函数体内：  
if(status){  
  async(['a'],function(a){  
    a.doSomething()  
  })  
}  
至此可以对由commonJS衍生出来的方案做出总结了。在浏览器端来设计模块加载机制，需要考虑依赖的问题。  
我们先把依赖分为两种，“强依赖” —— 肯定需要 和“弱依赖” —— 可能需要。  
对于强依赖，如果要性能优先，则考虑参照依赖前置的思想设计你的模块加载器，我个人也更推崇这个方案一些；如果考虑开发成本优先，则考虑按照依赖就近的思想设计你的模块加载器。  
对于弱依赖，只需要将弱依赖的部分改写到回调函数内即可。  
如果现在我要实现一个模块加载器，我会将强依赖前置，弱依赖采用异步回调函数的形式，其它的方法我认为都只是语法糖而已，仅此就够了。

25、请谈谈你都使用过哪些javascript模板

26、svg 与canvas的不同

SVG  
SVG 是一种使用 XML 描述 2D 图形的语言。  
SVG 基于 XML，这意味着 SVG DOM 中的每个元素都是可用的。您可以为某个元素附加 JavaScript 事件处理器。  
在 SVG 中，每个被绘制的图形均被视为对象。如果 SVG 对象的属性发生变化，那么浏览器能够自动重现图形。  
Canvas  
Canvas 通过 JavaScript 来绘制 2D 图形。  
Canvas 是逐像素进行渲染的。  
在 canvas 中，一旦图形被绘制完成，它就不会继续得到浏览器的关注。如果其位置发生变化，那么整个场景也需要重新绘制，包括任何或许已被图形覆盖的对象。  
27、localhost与sessionStorage的不同  
28、jQuery  Mobile 是什么  
jQuery Mobile 是创建移动 web 应用程序的框架。  
Jquery mobile是一个基于HTML5，拥有响应式网站特性，兼容所有主流移动设备平台的统一UI接口系统，与前端开发框架。可以运行在所有智能手机，平板电脑和桌面设备上。 jQuery Mobile秉承 “write less, do more”的原则到了一个新的高度：不需要为每一个移动设备或者操作系统单独开发应用，jQuery Mobile框架可以使您设计一个高度响应式的网站或应用运行于所有流行的智能手机，平板电脑和桌面系统。

27、localhost与sessionStorage的不同

28、jQuery Mobile 是什么

29、原生JS的window.onload与jquery的$(document).ready(function(){}) 有什么不同？

30、jquery与jQuery UI有什么区别？

jQuery UI 与 jquery 的主要区别是： 　  
　(1) jQuery是一个js库，主要提供的功能是选择器，属性修改和事件绑定等等。 　  
　(2) jQuery UI则是在jQuery的基础上，利用jQuery的扩展性，设计的插件。提供了一些常用的界面元素，诸如对话框、拖动行为、改变大小行为等等。 　  
　(3) jQuery本身注重于后台，没有漂亮的界面，而jQuery UI则补充了前者的不足，他提供了华丽的展示界面，使人更容易接受。既有强大的后台，又有华丽的前台。jQuery UI是jQuery插件，只不过专指由jQuery官方维护的UI方向的插件。

31、静态资源CDN如何使用

32、请尽可能详尽的解释AJXA的工作原理

33、ajax 是什么？Ajax的交互模型？同步和异步的区别？如何解决跨域问题？

34、请用jquery和原生js分别实现添加、移除、移动、复制、创建和查找DOM结点

一、创建元素节点  
1.1 原生JS创建元素节点  
document.createElement("p");  
1.2 jQuery创建元素节点  
$('<p></p>');`  
二、创建并添加文本节点  
2.1 原生JS创建文本节点  
`document.createTextNode("Text Content");  
通常创建文本节点和创建元素节点配合使用，比如：  
var textEl = document.createTextNode("Hello World.");  
var pEl = document.createElement("p");  
pEl.appendChild(textEl);  
2.2 jQuery创建并添加文本节点：  
var $p = $('<p>Hello World.</p>');  
三、复制节点  
3.1 原生JS复制节点:  
var newEl = pEl.cloneNode(true); `  
true和false的区别：  
true ：克隆整个'<p>Hello World.</p>'节点  
false：只克隆'<p></p>' ，不克隆文本Hello World.'  
3.2 jQuery复制节点  
$newEl = $('#pEl').clone(true);  
注意：克隆节点要避免`ID重复  
四、 插入节点  
4.1 原生JS向子节点列表的末尾添加新的子节点  
El.appendChild(newNode);  
原生JS在节点的已有子节点之前插入一个新的子节点：  
El.insertBefore(newNode, targetNode);  
4.2 在jQuery中，插入节点的方法比原生JS多的多  
在匹配元素子节点列表结尾添加内容  
$('#El').append('<p>Hello World.</p>');  
把匹配元素添加到目标元素子节点列表结尾  
$('<p>Hello World.</p>').appendTo('#El');  
在匹配元素子节点列表开头添加内容  
$('#El').prepend('<p>Hello World.</p>');  
把匹配元素添加到目标元素子节点列表开头  
$('<p>Hello World.</p>').prependTo('#El');  
在匹配元素之前添加目标内容  
$('#El').before('<p>Hello World.</p>');  
把匹配元素添加到目标元素之前  
$('<p>Hello World.</p>').insertBefore('#El');  
在匹配元素之后添加目标内容  
$('#El').after('<p>Hello World.</p>');  
把匹配元素添加到目标元素之后  
$('<p>Hello World.</p>').insertAfter('#El');  
五、删除节点  
5.1 原生JS删除节点  
El.parentNode.removeChild(El);  
5.2 jQuery删除节点  
$('#El').remove();  
六、替换节点  
6.1 原生JS替换节点  
El.repalceChild(newNode, oldNode);  
注意：oldNode必须是parentEl真实存在的一个子节点  
6.2 jQuery替换节点  
$('p').replaceWith('<p>Hello World.</p>');  
七、设置属性/获取属性  
7.1 原生JS设置属性/获取属性  
imgEl.setAttribute("title", "logo");  
imgEl.getAttribute("title");  
checkboxEl.checked = true;  
checkboxEl.checked;  
7.2 jQuery设置属性/获取属性:  
$("#logo").attr({"title": "logo"});  
$("#logo").attr("title");  
$("#checkbox").prop({"checked": true});  
$("#checkbox").prop("checked");

35、jquery怎样获取div的id为‘content’的文本内容。

$(“#content”).html

36、ajax请求的时候get和post方式的区别，并写出ajax的对象创建，发送，返回数据的 过程。

一个在url后面 一个放在虚拟载体里面  
有大小限制  
安全问题  
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

37、谈谈你对跨域访问的认识，写出两个实现跨域访问页面的的方法

38、用js或jQuery获取id为chuchujie的div里边的所有a标签，并把字体大小设置为18px； 同时把第三个a标签设置为块级元素，并把他的宽高分别设置为300px和500px

 var as = document.getElementById('chuanchuanjie').getElementsByTagName('a');  
        for(var i = 0,len = as.length;i<len;i++){  
            if(i==2){  
                as[i].style.display = 'block';  
                as[i].style.height = '500px';  
                as[i].style.width = '300px';  
            }  
            as[i].style.fontSize = '18px';  
        }

39、AMD（Modules/Asynchronous-Definition）、CMD（Common Module Definition）规范区 别？

40、documen.write和 innerHTML的区别

document.write是直接写入到页面的内容流，如果在写之前没有调用document.open, 浏览器会自动调用open。每次写完关闭之后重新调用该函数，会导致页面被重写。

innerHTML则是DOM页面元素的一个属性，代表该元素的html内容。你可以精确到某一个具体的元素来进行更改。如果想修改document的内容，则需要修改document.documentElement.innerElement。

innerHTML很多情况下都优于document.write，其原因在于其允许更精确的控制要刷新页面的那一个部分

41、[jQuery](http://lib.csdn.net/base/jquery" \t "http://blog.csdn.net/kuangruike/article/details/_blank" \o "jQuery知识库)与[jQuery](http://lib.csdn.net/base/jquery" \t "http://blog.csdn.net/kuangruike/article/details/_blank" \o "jQuery知识库) UI 有啥区别？

jQuery UI 与 jquery 的主要区别是： 　  
  
　(1) jQuery是一个js库，主要提供的功能是选择器，属性修改和事件绑定等等。 　  
　(2) jQuery UI则是在jQuery的基础上，利用jQuery的扩展性，设计的插件。提供了一些常用的界面元素，诸如对话框、拖动行为、改变大小行为等等。 　  
　(3) jQuery本身注重于后台，没有漂亮的界面，而jQuery UI则补充了前者的不足，他提供了华丽的展示界面，使人更容易接受。既有强大的后台，又有华丽的前台。jQuery UI是jQuery插件，只不过专指由jQuery官方维护的UI方向的插件。

42、[jQuery](http://lib.csdn.net/base/jquery" \t "http://blog.csdn.net/kuangruike/article/details/_blank" \o "jQuery知识库)的源码看过吗？能不能简单说一下它的实现原理？

jquery的外衣  
  
jquery是一个轻量级的JS框架，这点相信大部分人都听过，而jquery之所以有这样一个称呼，就是因为它悄悄披了一件外衣，将自己给隐藏了起来。  
  
//以下截取自jquery源码片段(function( window, undefined ) { /\* 源码内容 \*/})( window );  
  
上面这一小段代码来自于1.9.0当中jquery的源码，它是一个无污染的JS插件的标准写法，专业名词叫闭包。可以把它简单的看做是一个函数，与普通函数不同的是，这个函数没有名字，而且会立即执行，就像下面这样，会直接弹出字符串。  
  
(function( window, undefined ) { alert("Hello World!"); })( window );  
  
可以看出来这样写的直接效果，就相当于我们直接弹出一个字符串。但是不同的是，我们将里面的变量变成了局域变量，这不仅可以提高运行速度，更重要的是我们在引用jquery的JS文件时，不会因为jquery当中的变量太多，而与其它的JS框架的变量命名产生冲突。对于这一点，我们拿以下这一小段代码来说明。  
  
var temp = "Hello World!"; (function( window, undefined ) { var temp = "ByeBye World!"; })( window ); alert(temp);  
  
这段代码的运行结果是Hello而不是ByeBye，也就是说闭包中的变量声明没有污染到外面的全局变量，倘若我们去掉闭包，则最终的结果会是ByeBye，就像下面这样。  
  
var temp = "Hello World!";// (function( window, undefined ) { var temp = "ByeBye World!";// })( window ); alert(temp);  
  
由此就可以看出来，jquery的外衣就是这一层闭包，它是很重要的一个内容，是编写JS框架必须知道的知识，它可以帮助我们隐藏我们的临时变量，降低污染。  
  
jquery的背心  
  
刚才我们说了，jquery将自己声明的变量全部都用外衣遮盖起来了，而我们平时使用的Jquery和$，却是真真实实的全局变量，这个是从何而来，谜底就在jquery的某一行代码，一般是在文件的末尾。  
  
window.jQuery = window.$ = jQuery;  
  
这一句话将我们在闭包当中定义的jQuery对象导出为全局变量jQuery和$，因此我们才可以在外部直接使用jQuery和$。window是默认的JS上下文环境，因此将对象绑定到window上面，就相当于变成了传统意义上的全局变量，就像下面这一小段代码的效果一样。  
  
var temp = "Hello World!"; (function( window, undefined ) { var temp = "ByeBye World!"; window.temp = temp; })( window ); alert(temp);  
  
很明显，它的结果应该是ByeBye，而不是Hello。因为我们在闭包中导出了temp局部变量为全局变量，从而覆盖了第一行声明的全局变量temp。可以看出，就是通过导出的方式，jquery露出了自己的小背心。  
  
jquery的内裤  
  
  
内裤保护的是我们的核心器官，因此非常重要。那么jquery的内裤也一样，也是最核心的功能，就是选择器。而选择器简单理解的话，其实就是在DOM文档中，寻找一个DOM对象的工具。  
  
首先我们进入jquery源码中，可以很容易的找到jquery对象的声明，看过以后会发现，原来我们的jquery对象就是init对象。  
  
jQuery = function( selector, context ) { return new jQuery.fn.init( selector, context, rootjQuery ); }  
  
这里出现了jQuery.fn这样一个东西，它的由来可以在jquery的源码中找到，它其实代表的就是jQuery对象的原型。  
  
jQuery.fn = jQuery.prototype;jQuery.fn.init.prototype = jQuery.fn;  
  
这两句话，第一句把jQuery对象的原型赋给了fn属性，第二句把jQuery对象的原型又赋给了init对象的原型。也就是说，init对象和jQuery具有相同的原型，因此我们在上面返回的init对象，就与jQuery对象有一样的属性和方法。

43、jquery 中如何将数组转化为json字符串，然后再转化回来？

44、针对 jQuery 的优化方法？

1、总是使用#id去寻找element.  
 2、在Classes前面使用Tags  
 3、缓存jQuery对象  
 4、更好的利用链  
 5、使用子查询  
 6、限制直接对DOM操作  
 7、事件委托（又名：冒泡事件）  
 8、消除查询浪费  
 9、遵从$(windows).load  
 10、压缩JS

45、那些操作会造成内存泄漏？

严格意义上的内存泄露的原因只有一种：没有释放向系统申请的内存，因为不申请内存，就谈不上什么泄露，搞清楚内存泄露的原因，应当从汇编语言的角度考虑问题。  
当然没有释放内存的原因是多种的：  
有可能是你自己代码写的不好，忘记了释放自己代码里申请的内存，  
也有可能是你使用了一个写的不好的库，库本身有问题，这里说的库不仅仅是第三方库，甚至于各种语言的运行时库也有可能出现（再高的人都免不了出BUG），还甚至于操作系统的库，因为操作系统的BUG也多的很（当然系统一般情况不会出现这些低级的错误）。  
死循环不能说是内存泄露，概念上应称为死锁，死锁的确是有可能会导致内存无限量增长，但其与内存泄露有本质区别，尽管有时候它们导致的结果在内存层面是相同的。

46、JQuery一个对象可以同时绑定多个事件，这是如何实现的？

jQuery("#id").click(func1).mouseover(func2)//方法连写，func为方法的名字  
jQuery("#id").click(function(){//你的具体方法实现}),mouser(function(){//你的具体方法实现});  
jQuery("#id").bind("click mouseover",func)//两个事件中间有空格 ，func为方法的名字  
jQuery("#id").bind("load scroll",function(){//你的具体方法实现});

47、http状态码有那些？分别代表是什么意思？

状态码主要是用来看，就是当你的网站被收录，或好久没更新了，通过日志你看下搜索引擎抓取时的状态码是什么，是否正常。

例如：

成功2×× 成功处理了请求的状态码。

200 服务器已成功处理了请求并提供了请求的网页。

204 服务器成功处理了请求，但没有返回任何内容。

重定向3×× 每次请求中使用重定向不要超过 5 次。

301 请求的网页已永久移动到新位置。当URLs发生变化时，使用301代码。搜索引擎索引中保存新的URL。

302 请求的网页临时移动到新位置。搜索引擎索引中保存原来的URL。

304 如果网页自请求者上次请求后没有更新，则用304代码告诉搜索引擎机器人，可节省带宽和开销。

客户端错误4×× 表示请求可能出错，妨碍了服务器的处理。

400 服务器不理解请求的语法。

403 服务器拒绝请求。

404 服务器找不到请求的网页。服务器上不存在的网页经常会返回此代码。

410 请求的资源永久删除后，服务器返回此响应。该代码与 404（未找到）代码相似，但在资源以前存在而现在不存在的情况下，有时用来替代404 代码。如果资源已永久删除，应当使用 301 指定资源的新位置。

服务器错误5×× 表示服务器在处理请求时发生内部错误。这些错误可能是服务器本身的错误，而不是请求出错。

500 服务器遇到错误，无法完成请求。

503 服务器目前无法使用（由于超载或停机维护）。通常，这只是暂时状态。

48、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）

这个过程可以分为四个步骤：  
  
1当发送一个URL请求时，不管这个URL是web页面的URL还是web页面的每个资源的URL，浏览器都会开启一个线程来处理这个请求，同时在远程DNS服务器上启动一个DNS查询，这样就可以使得浏览器获得请求对应的IP地址了。  
2浏览器通过与远程web服务器TCP三次握手协商来建立一个TCP/IP链接。三个握手包含一个同步报文，一个同步-应答报文和一个应答报文，这三个报文在浏览器和服务器之间进行传递，该握手首先由客户端尝试建立起通信，而后服务器应答并接受客户端的请求，最后由由客户端发出该请求已经被接受的报文。  
3一旦TCP/IP链接建立，浏览器会通过该链接向远程服务器发送HTTP的GET请求。远程服务器找到资源并使用HTTP响应返回该资源，值为200的HTTP状态码表示一个正确的响应。  
 4.此时，web服务器提供资源服务，客户端开始下载资源。

49、ajax的readystate有5个状态 ，每个状态表示什么？

0 － （未初始化）还没有调用send()方法   
1 － （载入）已调用send()方法，正在发送请求   
2 － （载入完成）send()方法执行完成，已经接收到全部响应内容   
3 － （交互）正在解析响应内容   
4 － （完成）响应内容解析完成，可以在客户端调用了

50、[Ajax的工作原理](http://www.kanzhun.com/k-mianshitimu/529320.html)

Ajax的工作原理相当于在用户和服务器之间加了—个中间层，使用户操作与服务器响应异步化。这样把以前的一些服务器负担的工作转嫁到客户端，利于客户端闲置的处理能力来处理，减轻服务器和带宽的负担，从而达到节约ISP的空间及带宽租用成本的目的

51、[AJAX的优缺点都有什么？](http://www.kanzhun.com/k-mianshitimu/529319.html)

优点：减轻服务器的负担,按需取数据,最大程度的减少冗余请求  
局部刷新页面,减少用户心理和实际的等待时间,带来更好的用户体验  
基于xml标准化,并被广泛支持,不需安装插件等  
进一步促进页面和数据的分离  
缺点：[AJAX](https://www.baidu.com/s?wd=AJAX&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Ykrj9-rjFWP1cdmvc3mvF-0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnHc3PWRYPjRL" \t "_blank)大量的使用了javascript和[ajax](https://www.baidu.com/s?wd=ajax&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Ykrj9-rjFWP1cdmvc3mvF-0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnHc3PWRYPjRL" \t "_blank)引擎,这些取决于浏览器的支持.在编写的时候考虑对浏览器的兼容性.  
[AJAX](https://www.baidu.com/s?wd=AJAX&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Ykrj9-rjFWP1cdmvc3mvF-0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnHc3PWRYPjRL" \t "_blank)只是局部刷新,所以页面的后退按钮是没有用的.  
对流媒体还有移动设备的支持不是太好等

52、[如何开发一个JQuery插件？](http://www.kanzhun.com/k-mianshitimu/529315.html)

jquery插件一般有三种开发方式：  
通过$.extend()来扩展jQuery  
通过$.fn 向jQuery添加新的方法  
通过$.widget()应用jQuery UI的部件工厂方式创建

53、[Prototype如何实现页面局部定时刷新？](http://www.kanzhun.com/k-mianshitimu/529312.html)

54、[AJAX应用和传统Web应用有什么不同？](http://www.kanzhun.com/k-mianshitimu/529311.html)

用户需要点击”Submit”按钮来发送或者接受数据信息，然后等待服务器响应请求，页面重新加载。因为服务器每次都会返回一个新的页面， 所以传统的web应用有可能很慢而且用户交互不友好。使用AJAX技术， 就可以使Javascript通过XMLHttpRequest对象直接与服务器进行交互。通过HTTP Request， 一个web页面可以发送一个请求到web服务器并且接受web服务器返回的信息(不用重新加载页面)，展示给用户的还是通一个页面，用户感觉页面刷新，也看不到到Javascript后台进行的发送请求和接受响应。

55、jsonp的原理

上边有

56、jsonp的优缺点

JSONP的优缺点  
        1.优点  
                1.1它不像XMLHttpRequest对象实现的Ajax请求那样受到同源策略的限制，JSONP可以跨越同源策略；  
                1.2它的兼容性更好，在更加古老的浏览器中都可以运行，不需要XMLHttpRequest或ActiveX的支持  
                1.3在请求完毕后可以通过调用callback的方式回传结果。将回调方法的权限给了调用方。这个就相当于将controller层和view层终于分开了。我提供的jsonp服务只提供纯服务的数据，至于提供服务以 后的页面渲染和后续view操作都由调用者来自己定义就好了。如果有两个页面需要渲染同一份数据，你们只需要有不同的渲染逻辑就可以了，逻辑都可以使用同 一个jsonp服务。  
        2.缺点  
                2.1它只支持GET请求而不支持POST等其它类型的HTTP请求  
                2.2它只支持跨域HTTP请求这种情况，不能解决不同域的两个页面之间如何进行JavaScript调用的问题。  
                2.3 jsonp在调用失败的时候不会返回各种HTTP状态码。  
                2.4缺点是安全性。万一假如提供jsonp的服务存在页面注入漏洞，即它返回的javascript的内容被人控制的。那么结果是什么？所有调用这个 jsonp的网站都会存在漏洞。于是无法把危险控制在一个域名下…所以在使用jsonp的时候必须要保证使用的jsonp服务必须是安全可信的。

57、ajax请求时，如何解释json数据、

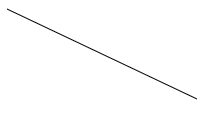
58、解释jsonp的原理，以及为什么不是真正的ajax

动态创建script标签，回调函数  
Ajax是页面无刷新请求数据操作

59、什么是Ajax和JSON，它们的优缺点。

[AJAX](https://www.baidu.com/s?wd=AJAX&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y3nAcLuHc3nHTsnjfYm1f0IAYqnWm3PW64rj0d0AP8IA3qPjfsn1bkrjKxmLKz0ZNzUjdCIZwsrBtEXh9GuA7EQhF9pywdQhPEUiqkIyN1IA-EUBtdnWbvPWDvrHT" \t "_blank) 是一种技术把 通俗的话 你可以当成是工具，它是你在不用刷新页面的时候 用来向后台交互的，  
而[JSON](https://www.baidu.com/s?wd=JSON&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y3nAcLuHc3nHTsnjfYm1f0IAYqnWm3PW64rj0d0AP8IA3qPjfsn1bkrjKxmLKz0ZNzUjdCIZwsrBtEXh9GuA7EQhF9pywdQhPEUiqkIyN1IA-EUBtdnWbvPWDvrHT" \t "_blank) 你可以想象成一个整理收纳袋 里面可以放很多东西 随便你 但是每个格格你贴了标签，然后有一个大家都知道的使用说明（实际就告诉大家（很多工具比如[JQ](https://www.baidu.com/s?wd=JQ&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y3nAcLuHc3nHTsnjfYm1f0IAYqnWm3PW64rj0d0AP8IA3qPjfsn1bkrjKxmLKz0ZNzUjdCIZwsrBtEXh9GuA7EQhF9pywdQhPEUiqkIyN1IA-EUBtdnWbvPWDvrHT" \t "_blank)和[JAVA](https://www.baidu.com/s?wd=JAVA&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y3nAcLuHc3nHTsnjfYm1f0IAYqnWm3PW64rj0d0AP8IA3qPjfsn1bkrjKxmLKz0ZNzUjdCIZwsrBtEXh9GuA7EQhF9pywdQhPEUiqkIyN1IA-EUBtdnWbvPWDvrHT" \t "_blank)代码）怎么去取你放里面的东西）

60、我们如何使用canvas来画一条简单的线。

定义Canvas区域获取访问canvas上下文区域绘制图形定义Canvas区域定义Canvas区域你需要使用下面的HTML代码，这定义了你能进行绘图的区域&#60;canvas id="mycanvas" width="600" height="500" style="border:1px solid #000000;"&#62;&#60;/canvas&#62;获取画布区域的访问在画布上进行绘图我们首先需要获取上下文区域的关联，下面是获取画布区域的代码。var c=document.getElementById("mycanvas");var ctx=c.getContext("2d");绘制图形现在一旦你获取了访问上下文，我们就可以开始在上下文中绘制了。首先调用&#8220;move&#8221;方法并从一个点开始，使用线条方法绘制线条然后使用stroke方法结束。ctx.moveTo(10,10);ctx.lineTo(200,100);ctx.stroke();完整的代码  
&#60;body&#160;&#160;onload="DrawMe();"&#62;  
&#60;canvas id="mycanvas" width="600" height="500" style="border:1px solid #000000;"&#62;&#60;/canvas&#62;  
&#60;/body&#62;  
&#60;script&#62;  
function DrawMe()  
{  
var c=document.getElementById("mycanvas");  
var ctx=c.getContext("2d");  
ctx.moveTo(10,10);  
ctx.lineTo(200,100);  
ctx.stroke();  
}  
你可以得到以下输出  


61、jQuery的事件委托方法on、live、delegate之间有什么区别？

62、一次完整的HTTP事务是怎样的一个过程？

域名解析 --> 发起TCP的3次握手 --> 建立TCP连接后发起http请求 --> 服务器响应http请求，浏览器得到html代码 --> 浏览器解析html代码，并请求html代码中的资源（如js、css、图片等） --> 浏览器对页面进行渲染呈现给用户

63、HTTPS是如何实现加密？

1，客户端将SSL 协议的[版本号](http://baike.baidu.com/view/421712.htm" \t "_blank)、加密算法的种类，产生的随机数A等信息传给服务器

2，服务器选择其中的一种组合作为加密方式，同时将自己的证书、公钥、另外一个随机数B一起传给客户端

3，客户端验证客户端返回的信息（包括证书，签名，域名等），验证成功，则生成对称加密密钥S，用公钥加密后返回给服务器

4，服务器用私钥将这段密钥解密。得到对称密钥S，并用此密钥加密一段握手消息返回给客户端

5，客户端收到握手消息，用对称密钥解密，验证成功，则握手成功。

64、异步ajax的优缺点都有什么。

优点：异步请求可以完全不影响用户的体验效果，无论请求的时间长或者短，用户都在专心的操作页面的其他内容，并不会有等待的感觉。  
缺点：

65、常用js框架有什么。Jq用过吗，jq的优点是什么。

jQuery是继prototype之後又一个优秀的Javascrīpt框架。其宗旨是——WRITE LESS,DO MORE，写更少的代码，做更多的事情。

它是轻量级的js库(压缩後只有21k) ，这是其它的js库所不及的，它兼容[CSS3](https://www.baidu.com/s?wd=CSS3&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1dbPHIBuW9hnvDLmWf4nAf0IAYqnWm3PW64rj0d0AP8IA3qPjfsn1bkrjKxmLKz0ZNzUjdCIZwsrBtEXh9GuA7EQhF9pywdQhPEUiqkIyN1IA-EUBtdrjRLPWR4njm" \t "_blank)，还兼容各种浏览器 （IE 6.0+, FF 1.5+, Safari 2.0+, Opera 9.0+）。

jQuery是一个快速的，简洁的javaScript库，使用户能更方便地处理HTML documents、events、实现动画效果，并且方便地为网站提供AJAX交互。

jQuery还有一个比较大的优势是，它的文档说明很全，而且各种应用也说得很详细，同时还有许多成熟的插件可供选择。

jQuery能够使用户的html页保持代码和html内容分离，也就是说，不用再在html里面插入一堆js来调用命令了，只需定义id即可。

66、JSONP是什么？它是如何实现跨域的？为什么它可以实现跨域？

JSONP(JSON with Padding)是JSON的一种“使用模式”，可用于解决主流浏览器的跨域数据访问的问题。  
1. 在客户端调用提供JSONP支持的URL Service，获取JSONP格式数据。  
  
比如客户想访问C:\Users\ADMINI~1\AppData\Local\Temp\%W@GJ$ACOF(TYDYECOKVDYB.pnghttp://www.yiwuku.com/myService.aspx?jsonp=callbackFunction  
  
假设客户期望返回JSON数据:["customername1","customername2"]  
  
那么真正返回到客户端的Script Tags: callbackFunction(["customername1","customername2"])  
  
可能的调用方式:  
  
2.在客户端写callbackfunction函数的实现  
  
3.页面展示  
  
4.最终page code  
体现：jquery下jsoncallback=？，其中“？”会自动替代function（data）函数。  
  
JSONP的最基本的原理是：动态添加一个<script>标签，而script标签的src属性是没有跨域的限制的。这样说来，这种跨域方式其实与ajax XmlHttpRequest协议无关了

67、如何解决跨域问题?

上边有

68、documen.write和 innerHTML的区别?

69、异步加载和延迟加载？

一、同步加载  
　　平常默认用的都是同步加载。如：  
<script src="[http://yourdomain.com/script.js](http://yourdomain.com/script.js" \t "_blank)"></script>   
　　同步模式又称阻塞模式，会阻止流览器的后续处理。停止了后续的文件的解析，执行，如图像的渲染。流览器之所以会采用同步模式，是因为加载的js文件中有对dom的操作，重定向，输出document等默认行为，所以同步才是最安全的。  
　　通常会把要加载的js放到body结束标签之前，使得js可在页面最后加载，尽量减少阻塞页面的渲染。这样可以先让页面显示出来。  
  
　　同步加载流程是瀑布模型，异步加载流程是并发模型。  
  
二、异步加载  
(function() {   
var s = document.createElement('script');   
s.type = 'text/javascript';   
s.async = true;   
s.src = '[http://yourdomain.com/script.js'](http://yourdomain.com/script.js'" \t "_blank);   
var x = document.getElementsByTagName('script')[0];   
 x.parentNode.insertBefore(s, x);   
})();  
  
异步加载也叫非阻塞模式加载，浏览器在下载js的同时，同时还会执行后续的页面处理。  
在script标签内，用js创建一个script元素并插入到document中，这种就是异步加载js文件了

70、模块化怎么做？

1。做三种不同大小的，分别打包，然后用专业打包程序打包一起。  
2。在README中著明

71、.call() 和 .apply() 的区别？

**call方法:**   
语法：call(thisObj，Object)  
定义：调用一个对象的一个方法，以另一个对象替换当前对象。  
说明：  
call 方法可以用来代替另一个对象调用一个方法。call 方法可将一个函数的对象上下文从初始的上下文改变为由 thisObj 指定的新对象。   
如果没有提供 thisObj 参数，那么 Global 对象被用作 thisObj。   
  
**apply方法：**   
语法：apply(thisObj，[argArray])  
定义：应用某一对象的一个方法，用另一个对象替换当前对象。   
说明：   
如果 argArray 不是一个有效的数组或者不是 arguments 对象，那么将导致一个 TypeError。   
如果没有提供 argArray 和 thisObj 任何一个参数，那么 Global 对象将被用作 thisObj， 并且无法被传递任何参数。

72、jquery 中如何将数组转化为json字符串，然后再转化回来？

73、那些操作会造成内存泄漏？

上边有

74、JQuery一个对象可以同时绑定多个事件，这是如何实现的？

**第四阶段**

1. 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

、为了让网页更快的加载，你会如何处理你的css文件和js文件，以及图片文件，页面性能优化你还知道哪些方法。

3、Node.js的适用场景

4、对Node的优点和缺点提出了自己的看法

（优点）因为Node是基于事件驱动和无阻塞的，所以非常适合处理并发请求，

因此构建在Node上的代理服务器相比其他技术实现（如Ruby）的服务器表现要好得多。

此外，与Node代理服务器交互的客户端代码是由javascript语言编写的，

因此客户端和服务器端都用同一种语言编写，这是非常美妙的事情。

（缺点）Node是一个相对新的开源项目，所以不太稳定，它总是一直在变，

而且缺少足够多的第三方库支持。看起来，就像是Ruby/Rails当年的样子。

5、node.js、Mongodb、npm、MVVM、MEAN、three.js

6、使用Promises 而非回调（callbacks）优缺点是什么

7、请使用任意一种mvvm框架演示如何实现双向绑定

### (1).视图变动的监听

MVVM框架都是通过相应的指令，在HTML中声明式的标记出需要监听的DOM节点。本文实现中，我们主要涉及到两个指令： **foio-controller** 、 **foio-model** 以及一个表达式{{}}。 比如：

**<input type="text" foio-model="nickname">**

上述指令foio-model，声明将View中的input的变动通知到Model中的nickname。通过对的视图节点(input)注册监听函数就可以得到视图(input)的变动了。

**//对视图中的input节点注册input事件监听函数**

**var elem = document.querySelector('input');**

**if (elem.addEventListener) {**

**elem.addEventListener('input', callback, false);**

**} else {**

**elem.attachEvent('oninput', callback);**

**}**

### (2).模型变动的监听

对模型变动的监听可以通过ECMAScript5中的API实现。

**Object.defineProperty(obj, prop, descriptor)**

可以通过该API为对象添加一个属性，并设置该属性的gett函数和set函数，在访问属性时会触发相应的get函数和set函数。

**var air = {};**

**Object.defineProperty(air, 'temperature', {**

**get: function() {**

**console.log('get!');**

**},**

**set: function(value) {**

**console.log('set!');**

**}**

**});**

**air.temperature = 15; //output: set!**

**air.temperature; //outpu: get!**

我们可以在set函数中得到模型的变动，并将相关变动通知到ViewModel。

## 2.总体实现

MVVM的主要流程包括(View)视图扫描、(Model)模型构建、以及关联视图和模型(ViewModel)

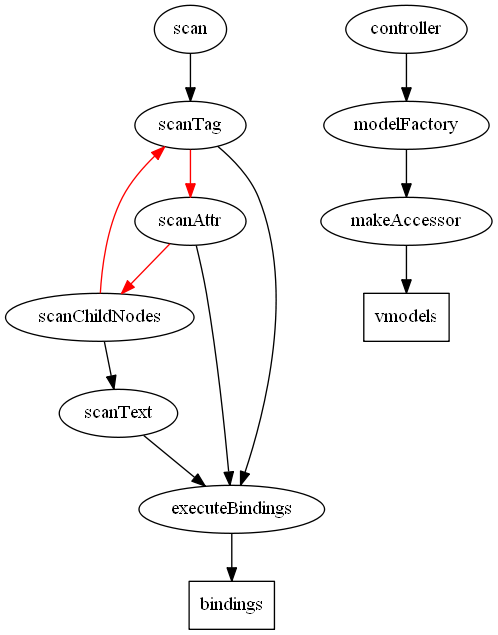
### (1)View(视图)扫描

处理View(视图)必然涉及到对DOM结构的扫描，通过扫描抽取指令(本文只有三种指令，foio-controller、foio-model、)；并对相应的节点进行如下处理：

**绑定通知函数，用于在视图更新时通知ViewModel**

**绑定更新函数，用于在模型更新时通过该函数更新视图**

针对不同的节点类型，这些通知函数和更新函数都是预先定义好的，存储在 **directives** 结构中。在节点扫描过程中，当遇到指令时，就通过executeBindings函数对相应的节点进行绑定处理。流程图如下：



### (2)Model(模型)构建

而对Model的处理也主要是注册监听函数，用于在Model变化时得到通知，如上图所示。controller中的每一个变量都通过 **Object.defineProperty(obj, prop, descriptor)** 定义到Model上，其中descriptor上的get函数可以用于搜集依赖，而set函数则用于通知依赖于该Model的视图进行更新。

**var descriptor = {**

**var dependencyList = [];**

**get: function() {**

**//搜集依赖**

**dependencyList.push(this);**

**return value;**

**},**

**set: function(newVal) {**

**if (oldVal === newVal) {**

**return;**

**}**

**oldVal = newVal;**

**//通知依赖于该Model的视图进行更新**

**for (var dependIdx in dependencyList) {**

**dependencyList[dependIdx].updateView(newVal);**

**}**

**}**

**}**

### (3)关联模型和视图

View(视图)扫描的结果是一个元素集合

**bindings = [**

**{**

**type: type, //指令类型**

**element: elem, //DOM节点**

**expr: value, //绑定的变量名称**

**},**

**{...}**

**]**

而Model(模型)构建的结果也是一个集合：

**vmodels = {**

**controller1: {**

**expr1: value1,**

**expr2: value2,**

**binder: {expr1: function(){},expr2:function(){}}**

**},**

**controller1: {...}**

**}**

通过executeBindings函数，将视图和模型关联起来。

**function executeBindings(bindings, vmodels) {**

**for (var i = 0, binding; (binding = bindings[i++]);) {**

**binding.vmodels = vmodels;**

**directives[binding.type](binding);**

**};**

**}**

每一种指令都有不同的初始化函数，比如针对 **foio-model** 指令，当DOM节点为input类型时，初始化函数做了三件事：

**监听input和DOMAutoComplete事件**

**注册对模型的依赖**

**提供更新该DOM节点的方法**

详细代码如下：

**directives['model']={**

**switch (binding.xtype) {**

**case "input":**

**//绑定input事件**

**binding.bound('input', updateVModel);**

**//绑定DOMAutoComplete事件**

**binding.bound('DOMAutoComplete', updateVModel);**

**//注册对模型的依赖**

**elem.value = closetVmodel.binder[binding.expr].apply(binding);**

**//更新该DOM节点的方法**

**binding.updateView = function(newVal) {**

**elem.value = newVal;**

**};**

**break;**

**}**

**}**

至此我们实现了一个基本的MVVM框架了，虽然只有三个指令，但是基本能够说明如何设计并实现一个MVVM框架了。

8、对Node的有点和缺点提出自己的看法

9、比较一下jQuery,Angular.js,React.js 等框架的优缺点

10、介绍下let，var，和const的区别

使用var声明的变量，其作用域为该语句所在的函数内，且存在变量提升现象；  
使用let声明的变量，其作用域为该语句所在的代码块内，不存在变量提升；  
使用const声明的是常量，在后面出现的代码中不能再修改该常量的值。

11、有了解ES6么？如何了解的？关于使用？

12、对react,vue，或者angular有过接触么？他们之前的差异你能简单说下？

1.vuejs非常小清新，小清新不代表做不了复杂的东西，比如官方的这个demo就不错：  
2.reactjs代码量最多，因为它既要管理UI逻辑，又要操心dom的渲染。  
3.angularjs渲染快，就是量大（相对于vue.js）。

13、用得比较多的前端框架是哪些?有看过源码么？有哪些看过源码啊？

14、说说 React 中的virtual dom原理？

最好不要把他看成virtual dom，其实就是内存里的一大堆原始的javascript对象，和dom一样，维持着一个树形的结构，这颗树和dom没关系，你操作这些原始的对象，virtual dom的代码根据一些对应来操作dom

15、angular数据绑定采用什么机制？远离是什么？

ng-model 双向绑定

16、eval是做什么的。

eval()函数，这个函数可以把一个字符串当作一个JavaScript表达式一样去执行它。

17、React为什么这么火？

react 和传统的web 2.0前端的确有点不同， 它的render 方式更像[服务器端](https://www.baidu.com/s?wd=%E6%9C%8D%E5%8A%A1%E5%99%A8%E7%AB%AF&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YzPHw-nHcznWf1PW7bnWfk0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnHf4rjDvnjb1njbdrHfknjT1rf" \t "_blank)render，就是  
1. 准备好所有数据（e.g. 通过ajax ..）  
2. 生成整个数据块virtual dom  
不但Fetch Data(e.g. 列表)是这样， Modify Data(e.g. 创建， 删除) 也是这样。  
  
按传统的方法，'Get list' and 'Delete item' 的 render(step2) 是不同的  
Get List:  
Step1. ajax call 去获取数据（Array）  
Step2. 从得到的Array生成HTML

18、什么是MVVM框架？

MVVM在概念上是真正将页面与数据逻辑分离的模式，它把数据绑定工作放到一个JS里去实现，而这个JS文件的主要功能是完成数据的绑定，即把model绑定到UI的元素上。

19、MVVM框架的优点

在MVC里，View是可以直接访问Model的！从而，View里会包含Model信息，不可避免的还要包括一些[业务逻辑](https://www.baidu.com/s?wd=%E4%B8%9A%E5%8A%A1%E9%80%BB%E8%BE%91&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1d-nhRdrAcvnHDvrHcYPvn10ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPWD3PWbknHTsP101rHmznHnz" \t "_blank)。 MVC模型关注的是Model的不变，所以，在MVC模型里，Model不依赖于View，但是 View是依赖于Model的。不仅如此，因为有一些[业务逻辑](https://www.baidu.com/s?wd=%E4%B8%9A%E5%8A%A1%E9%80%BB%E8%BE%91&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1d-nhRdrAcvnHDvrHcYPvn10ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPWD3PWbknHTsP101rHmznHnz" \t "_blank)在View里实现了，导致要更改View也是比较困难的，至少那些[业务逻辑](https://www.baidu.com/s?wd=%E4%B8%9A%E5%8A%A1%E9%80%BB%E8%BE%91&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1d-nhRdrAcvnHDvrHcYPvn10ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPWD3PWbknHTsP101rHmznHnz" \t "_blank)是无法重用的。  
  
　　MVVM在概念上是真正将页面与数据逻辑分离的模式，它把[数据绑定](https://www.baidu.com/s?wd=%E6%95%B0%E6%8D%AE%E7%BB%91%E5%AE%9A&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1d-nhRdrAcvnHDvrHcYPvn10ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPWD3PWbknHTsP101rHmznHnz" \t "_blank)工作放到一个JS里去实现，而这个JS文件的主要功能是完成数据的绑定，即把model绑定到UI的元素上。  
  
　　有人做过测试：使用Angular（MVVM）代替Backbone（MVC）来开发，代码可以减少一半。  
  
　　此外，MVVM另一个重要特性，双向绑定。它更方便你同时维护页面上都依赖于某个字段的N个区域，而不用手动更新它们。

20、使用原生js实现ajax

var Ajax={  
    get: function (url,fn){  
        var obj=new XMLHttpRequest();  // XMLHttpRequest对象用于在后台与服务器交换数据            
        obj.open('GET',url,true);  
        obj.onreadystatechange=function(){  
            if (obj.readyState == 4 && obj.status == 200 || obj.status == 304) { // readyState==4说明请求已完成  
                fn.call(this, obj.responseText);  //从服务器获得数据  
            }  
        };  
        obj.send(null);  
    },  
    post: function (url, data, fn) {  
        var obj = new XMLHttpRequest();  
        obj.open("POST", url, true);  
        obj.setRequestHeader("Content-type", "application/x-www-form-urlencoded"); // 发送信息至服务器时内容编码类型  
        obj.onreadystatechange = function () {  
            if (obj.readyState == 4 && (obj.status == 200 || obj.status == 304)) {  // 304未修改  
                fn.call(this, obj.responseText);  
            }  
        };  
        obj.send(data);  
    }  
}

21、ng-if 跟 ng-show/hide 的区别有哪些？

    实现原理方面：ng-show/ng-hide是通过修改CSS样式方式控制元素显示与隐藏，对应的DOM元素会一直存在于当前页面中，而ng-if根据表达式的值动态的在当前的页面中添加删除页面元素。如果赋值表达式的值为false，那么这个元素就会从页面中删除，否则会添加一个元素。ng-if创建元素时用的是被它编译后的代码，如果ng-if内部的代码被其它方式修改过，那么修改只会对本次展现有效，页面元素重新渲染后修改效果会消失，而ng-show/ng-hide则能够保留dom元素上次修改后的状态。

22、ng-repeat迭代数组的时候，如果数组中有相同值，会有什么问题，如何解决？

23、ng-click 中写的表达式，能使用 JS 原生对象上的方法吗？

24、angular 的数据绑定采用什么机制？详述原理

AngularJS的双向数据绑定采用脏检查（dirty-checking）机制。ng只有在指定事件触发后，才进入 $digest cycle ：   
- DOM事件，譬如用户输入文本，点击按钮等。( ng-click )   
- XHR响应事件 ( $http )   
- 浏览器Location变更事件 ( $location )   
- Timer事件( $timeout , $interval )   
- 执行 $digest() 或 $apply()

25、详述 angular 的 “依赖注入”

在软件工程中，依赖注入是实现控制反转的一种软件设计模式，一个依赖是一个被其他对象(client)调用的对象（服务）,注入则是将被依赖的对象(service)实例传递给依赖对象(client)的行为。将 被依赖的对象传给依赖者，而不需要依赖者自己去创建或查找所需对象是DI的基本原则。 依赖注入允许程序设计遵从依赖倒置原则（简单的说就是要求对抽象进行编程，不要对实现进行编程，这样就降低了客户与实现模块间的耦合） 调用者(client)只需知道服务的接口，具体服务的查找和创建由注入者(injector)负责处理并提供给client，这样就分离了服务和调用者的依赖，符合低耦合的程序设计原则。

26、angular 的缺点有哪些？

27、angular 应用常用哪些路由库，各自的区别是什么？

28、如何用Node监听80端口

29、你最喜欢的HTTP框架以及原因