**2018最新Web前端经典面试试题及答案-史上最全前端面试题(含答案)**

近期总结一一些面试题 都是企业的面试题笔记题

感觉薪资10k下的都会出笔试题

特别高的薪资都是直接技术面试或者是 现场编程

总结很多人的面试题，后期会对于单个知识点再说笔记详细讲解。

部分都是百度的答案，不是特全面的，可以自己找下

**红色为常见面试题**

=============================================================

**前端面试题：**

**1. 一个200\*200的div在不同分辨率屏幕上下左右居中，用css实现**

div   
{   
position:absolute;   
width:200px;   
height:200px;   
top:50%;   
left:50%;   
margin-left:-50px;   
height:-50px;   
z-index:1000;   
}

**2. 写一个左中右布局占满屏幕，其中左右两块是固定宽度200 ，中间自适应宽，要求先加载中间块，请写出结构及样式：**

 <body>

        <h3>实现三列宽度自适应布局</h3>

        <div id = "left">我是左边</div>

        <div id = "right">我是右边</div>

        <div id = "center">我是中间</div>

</body>

html,body{ margin: 0px;width: 100%; }

h3{height: 100px;margin:20px 0 0;}

#left,#right{width: 200px;height: 200px; background-color: #ffe6b8;position: absolute;top:120px;}

#left{left:0px;}

#right{right: 0px;}

#center{margin:2px 210px ;background-color: #eee;height: 200px; }

**3. 阐述清楚浮动的几种方式（常见问题）**

**1.父级div定义 height**

原理：父级div手动定义height，就解决了父级div无法自动获取到高度的问题。   
优点：简单、代码少、容易掌握   
缺点：只适合高度固定的布局，要给出精确的高度，如果高度和父级div不一样时，会产生问题

**2.父级div定义 overflow:hidden**

原理：必须定义width或zoom:1，同时不能定义height，使用overflow:hidden时，浏览器会自动检查浮动区域的高度   
优点：简单、代码少、浏览器支持好

4. **结尾处加空div标签 clear:both**

原理：添加一个空div，利用css提高的clear:both清除浮动，让父级div能自动获取到高度   
优点：简单、代码少、浏览器支持好、不容易出现怪问题

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

**5. 解释css sprites ，如何使用？**

CSS Sprites其实就是把网页中一些背景图片整合到一张图片文件中，再利用CSS的“background-image”，“background- repeat”，“background-position”的组合进行背景定位，background-position可以用数字能精确的定位出[背景图片](http://baike.baidu.com/view/950110.htm)的位置。

CSS Sprites为一些大型的网站节约了带宽，让提高了用户的加载速度和用户体验，不需要加载更多的图片

**6. 如何用原生js给一个按钮绑定两个onclick事件？**

Var  btn=document.getElementById(‘btn’);

//事件监听 绑定多个事件

var btn4 = document.getElementById("btn4");

btn4.addEventListener("click",hello1);

btn4.addEventListener("click",hello2);

function hello1(){

 alert("hello 1");

}

function hello2(){

 alert("hello 2");

}

**7. 拖拽会用到哪些事件**

· dragstart:拖拽开始时在被拖拽元素上触发此事件,监听器需要设置拖拽所需数据,从操作系统拖拽文件到浏览器时不触发此事件.

· dragenter:拖拽鼠标进入元素时在该元素上触发,用于给拖放元素设置视觉反馈,如高亮

· dragover:拖拽时鼠标在目标元素上移动时触发.监听器通过阻止浏览器默认行为设置元素为可拖放元素.

· dragleave:拖拽时鼠标移出目标元素时在目标元素上触发.此时监听器可以取消掉前面设置的视觉效果.

· drag:拖拽期间在被拖拽元素上连续触发

· drop:鼠标在拖放目标上释放时,在拖放目标上触发.此时监听器需要收集数据并且执行所需操作.如果是从操作系统拖放文件到浏览器,需要取消浏览器默认行为.

· dragend:鼠标在拖放目标上释放时,在拖拽元素上触发.将元素从浏览器拖放到操作系统时不会触发此事件.

**8. 请列举jquery中的选择器：**

**9. Javascript中的定时器有哪些？他们的区别及用法是什么？**

setTimeout 只执行一次  
setInterval 会一直重复执行

**9.请描述一下 cookies sessionStorage和localstorage区别**

**相同点：**都存储在客户端  
**不同点：**1.存储大小

· cookie数据大小不能超过4k。

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

2.有效时间

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

· sessionStorage  数据在当前浏览器窗口关闭后自动删除。

· cookie          设置的cookie过期时间之前一直有效，即使窗口或浏览器关闭

3. 数据与服务器之间的交互方式

· cookie的数据会自动的传递到服务器，服务器端也可以写cookie到客户端

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

**10.计算一个数组arr所有元素的和**

var arr1=[1,2,3,4,5,6,7,8,9];

var sum1=0;

for (var i=0;i<=arr1.length;i++) {

if (typeof arr1[i]=="number") {

sum1+=arr1[i];

}

}

document.write(sum1);

//====================================

function sum2(arr){

var all=0;

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

if (typeof arr[i]=="number") {

all+=arr[i];

}

}

return all;

}

document.write(sum2([1,2,3,4]));

**11.编写一个方法去掉数组里面 重复的内容  var arr=[1,2,3,4,5,1,2,3]**

[**一个数组去重的简单实现**](http://www.cnblogs.com/xiangru0921/p/6513511.html)

var arr = ['abc','abcd','sss','2','d','t','2','ss','f','22','d'];

//定义一个新的数组

var s = [];

//遍历数组

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

    if(s.indexOf(arr[i]) == -1){  //判断在s数组中是否存在，不存在则push到s数组中

        s.push(arr[i]);

    }

}

console.log(s);

//输出结果：["abc", "abcd", "sss", "2", "d", "t", "ss", "f", "22"]

方法二：用sort()  然后相邻比较也可以实现

**12.document.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。

**13.ajax的步骤**

**什么是ajax?**

ajax(异步javascript xml) 能够刷新局部网页数据而不是重新加载整个网页。

如何使用ajax?

第一步，创建xmlhttprequest对象，var xmlhttp =new XMLHttpRequest（);XMLHttpRequest对象用来和服务器交换数据。

var xhttp;

if (window.XMLHttpRequest) {

//现代主流浏览器

xhttp = new XMLHttpRequest();

} else {

// 针对浏览器，比如IE5或IE6

xhttp = new ActiveXObject("Microsoft.XMLHTTP");

}

第二步，使用xmlhttprequest对象的open（）和send（）方法发送资源请求给服务器。

**第三步**，使用xmlhttprequest对象的responseText或responseXML属性获得服务器的响应。

**第四步**，onreadystatechange函数，当发送请求到服务器，我们想要服务器响应执行一些功能就需要使用onreadystatechange函数，每次xmlhttprequest对象的readyState发生改变都会触发onreadystatechange函数

**14.xml和json的区别，请用四个词语来形容**

·  JSON相对于XML来讲，数据的体积小，传递的速度更快些

·  JSON与JavaScript的交互更加方便，更容易解析处理，更好的数据交互

·  XML对数据描述性比较好；

·  JSON的速度要远远快于XML

**15.清楚浮动的方法？(多次出现在面试题)**

**1.父级div定义 height**  
原理：父级div手动定义height，就解决了父级div无法自动获取到高度的问题。   
优点：简单、代码少、容易掌握   
  
缺点：只适合高度固定的布局，要给出精确的高度，如果高度和父级div不一样时，会产生问题

**2，结尾处加空div标签 clear:both**

原理：添加一个空div，利用css提高的clear:both清除浮动，让父级div能自动获取到高度   
优点：简单、代码少、浏览器支持好、不容易出现怪问题   
缺点：不少初学者不理解原理；如果页面浮动布局多，就要增加很多空div，让人感觉很不好  
**3，父级div定义 伪类:after 和 zoom**

原理：IE8以上和非IE浏览器才支持:after，原理和方法2有点类似，zoom(IE转有属性)可解决ie6,ie7浮动问题   
优点：浏览器支持好、不容易出现怪问题（目前：大型网站都有使用，如：腾迅，网易，新浪等等）   
缺点：代码多、不少初学者不理解原理，要两句代码结合使用才能让主流浏览器都支持  
**4，父级div定义 overflow:hidden**

原理：必须定义width或zoom:1，同时不能定义height，使用overflow:hidden时，浏览器会自动检查浮动区域的高度   
优点：简单、代码少、浏览器支持好   
缺点：不能和position配合使用，因为超出的尺寸的会被隐藏。

**16.box-sizing常用的属性有哪些？分别有什么作用？**

**属性值**

· box-sizing：content-box

· box-sizing：border-box

· box-sizing：inherit

**content-box**

· 这是box-sizing的默认属性值

· 是CSS2.1中规定的宽度高度的显示行为

· 在CSS中定义的宽度和高度就对应到元素的内容框

· 在CSS中定义的宽度和高度之外绘制元素的内边距和边框

**border-box**

· 在CSS中微元素设定的宽度和高度就决定了元素的边框盒

· 即为元素在设置内边距和边框是在已经设定好的宽度和高度之内进行绘制

· CSS中设定的宽度和高度减去边框和内间距才能得到元素内容所占的实际宽度和高度

(Q1)box-sizing: content-box|border-box|inherit;  
(Q2)content-box:宽度和高度分别应用到元素的内容框。在宽度和高度之外绘制元素的内边距和边框(元素默认效果)。  
border-box:元素指定的任何内边距和边框都将在已设定的宽度和高度内进行绘制。通过从已设定的宽度和高度分别减去边框和内边距才能得到内容的宽度和高度。

**17.css选择器有哪些，选择器的权重的优先级**

**选择器类型**

**1、ID　　#id**

**2、class　　.class**

**3、标签　　p**

**4、通用　　\***

**5、属性　　[type="text"]**

**6、伪类　　：hover**

**7、伪元素　　::first-line**

**8、子选择器、相邻选择器**

**三、权重计算规则**

1. **第一等：代表内联样式，如: style=””，权值为1000。**

2. **第二等：代表ID选择器，如：#content，权值为0100。**

3. **第三等：代表类，伪类和属性选择器，如.content，权值为0010。**

4. **第四等：代表类型选择器和伪元素选择器，如div p，权值为0001。**

5. **通配符、子选择器、相邻选择器等的。如\*、>、+,权值为0000。**

6. **继承的样式没有权值。**

**18. 块级元素水平垂直居中的方法有哪些（三个方法）**

让div等块级元素水平和垂直都居中，即永远处于屏幕的正中央，当我们做如登录块时非常有用！

　实现一、原理：要让div等块级元素水平和垂直居中，必需知道该div等块级元素的宽度和高度，然后设置位置为绝对位置，距离页面窗口左边框和上边框的距离设置为50%，这个50%就是指页面窗口的宽度和高度的50%，最后将该div等块级元素分别左移和上移，左移和上移的大小就是该div等块级元素宽度和高度的一半。

   CSS代码：

.mycss{

   width:300px;

   height:200px;

   position:absolute;

   left:50%;

   top:50%;

   margin:-100px 0 0 -150px }

 实现二原理：利用CSS的margin设置为auto让浏览器自己帮我们水平和垂直居中。

   CSS代码：

.mycss{

    position: absolute;

    left: 0px;

    right: 0;

    top: 0;

    bottom: 0;

    margin: auto;

    height: 200px;

    width: 300px;

}

jQuery实现水平和垂直居中

　原理：jQuery实现水平和垂直居中的原理就是通过jQuery设置div等块级元素的CSS，获取div等块级元素的左、上的边距偏移量，边距偏移量的算法就是用页面窗口 的宽度减去该div等块级元素的宽度，得到的值再除以2即左偏移量，右偏移量算法相同。注意div等块级元素的CSS设置要在resize()方法中完成，就是每次改变窗口大 小时，都要执行设置div等块级元素的CSS。

jquery代码：

$(window).resize(function(){

    $(".myblock").css({

        position: "absolute",

        left: ($(window).width() - $(".myblock").outerWidth())/2,

        top: ($(window).height() - $(".myblock").outerHeight())/2     });

});

此外在页面载入时，就需要调用resize()方法

$(function(){

    $(window).resize();

});

**19.三个盒子，左右定宽，中间自适应有几种方法**

**第一种：左右侧采用浮动 中间采用margin-left 和 margin-right 方法。**

代码如下：

<div style="width:100%; margin:0 auto;">

       <div style="width:200px; float:right; background-color:#960">这是右侧的内容 固定宽度</div>

       <div style="width:150px; float:left; background:#6FF">这是左侧的内容 固定宽度</div>

       <div style="margin-left:150px;margin-right:200px; background-color:#9F3">中间内容，自适应宽度</div>

    </div>

第二种：左右两侧采**用绝对定位** 中间同样采用margin-left margin-right方法：

第三种负的margin

使用这种方法就稍微复杂了一些了，使用的是负的margin值，而且html标签也增加了，先来看其代码吧：

<div id="main">

 <div id="mainContainer">main content</div></div><div id="left">

 <div id="leftContainer" class="inner">left content</div></div><div id="right">

 <div id="rightContainer" class="inner">right</div></div>

#main {

 float: left;

 width: 100%;

}

#mainContainer {

 margin: 0 230px;

 height: 200px;

 background: green;

}

#left {

 float: left;

 margin-left: -100%;

 width: 230px}

#right {

 float: left;

 margin-left: -230px;

 width: 230px;

}

#left .inner,

#right .inner {

 background: orange;

 margin: 0 10px;

 height: 200px;

}

**20.js有几种数据类型，其中基本数据类型有哪些**

五种基本类型: Undefined、Null、Boolean、Number和String。

1中复杂的数据类型————Object，Object本质上是由一组无序的名值对组成的。

Object、Array和Function则属于引用类型

**21.**[**undefined 和 null 区别**](http://www.cnblogs.com/ly0612/p/6696982.html)

null： Null类型，代表“空值”，代表一个空对象指针，使用typeof运算得到 “object”，所以你可以认为它是一个特殊的对象值。

undefined： Undefined类型，当一个声明了一个变量未初始化时，得到的就是undefined。

null是javascript的**关键字，**可以认为是对象类型，它是一个空对象指针，和其它语言一样都是代表“空值”，不过 undefined 却是javascript才有的。undefined是在ECMAScript第三版引入的，为了区分空指针对象和未初始化的变量，它是一个预定义的**全局变量**。没有返回值的函数返回为undefined，没有实参的形参也是undefined。

javaScript权威指南： null 和 undefined 都表示“值的空缺”，你可以认为undefined是表示系统级的、出乎意料的或类似错误的值的空缺，而null是表示程序级的、正常的或在意料之中的值的空缺。

**22.http 和 https 有何区别？如何灵活使用？**

http是HTTP协议运行在TCP之上。所有传输的内容都是明文，客户端和服务器端都无法验证对方的身份。

https是HTTP运行在SSL/TLS之上，SSL/TLS运行在TCP之上。所有传输的内容都经过加密，加密采用对称加密，但对称加密的密钥用服务器方的证书进行了非对称加密。此外客户端可以验证服务器端的身份，如果配置了客户端验证，服务器方也可以验证客户端的身份

**23.常见的HTTP状态码**

**2开头 （请求成功）表示成功处理了请求的状态代码。**

200   （成功）  服务器已成功处理了请求。 通常，这表示服务器提供了请求的网页。   
201   （已创建）  请求成功并且服务器创建了新的资源。   
202   （已接受）  服务器已接受请求，但尚未处理。   
203   （非授权信息）  服务器已成功处理了请求，但返回的信息可能来自另一来源。   
204   （无内容）  服务器成功处理了请求，但没有返回任何内容。   
205   （重置内容） 服务器成功处理了请求，但没有返回任何内容。  
206   （部分内容）  服务器成功处理了部分 GET 请求。

**3开头 （请求被重定向）表示要完成请求，需要进一步操作。 通常，这些状态代码用来重定向。**

300   （多种选择）  针对请求，服务器可执行多种操作。 服务器可根据请求者 (user agent) 选择一项操作，或提供操作列表供请求者选择。   
301   （永久移动）  请求的网页已永久移动到新位置。 服务器返回此响应（对 GET 或 HEAD 请求的响应）时，会自动将请求者转到新位置。  
302   （临时移动）  服务器目前从不同位置的网页响应请求，但请求者应继续使用原有位置来进行以后的请求。  
303   （查看其他位置） 请求者应当对不同的位置使用单独的 GET 请求来检索响应时，服务器返回此代码。  
304   （未修改） 自从上次请求后，请求的网页未修改过。 服务器返回此响应时，不会返回网页内容。   
305   （使用代理） 请求者只能使用代理访问请求的网页。 如果服务器返回此响应，还表示请求者应使用代理。   
307   （临时重定向）  服务器目前从不同位置的网页响应请求，但请求者应继续使用原有位置来进行以后的请求。

**4开头 （请求错误）这些状态代码表示请求可能出错，妨碍了服务器的处理。**

400   （错误请求） 服务器不理解请求的语法。   
401   （未授权） 请求要求身份验证。 对于需要登录的网页，服务器可能返回此响应。   
403   （禁止） 服务器拒绝请求。  
404   （未找到） 服务器找不到请求的网页。  
405   （方法禁用） 禁用请求中指定的方法。   
406   （不接受） 无法使用请求的内容特性响应请求的网页。   
407   （需要代理授权） 此状态代码与 401（未授权）类似，但指定请求者应当授权使用代理。  
408   （请求超时）  服务器等候请求时发生超时。   
409   （冲突）  服务器在完成请求时发生冲突。 服务器必须在响应中包含有关冲突的信息。   
410   （已删除）  如果请求的资源已永久删除，服务器就会返回此响应。   
411   （需要有效长度） 服务器不接受不含有效内容长度标头字段的请求。   
412   （未满足前提条件） 服务器未满足请求者在请求中设置的其中一个前提条件。   
413   （请求实体过大） 服务器无法处理请求，因为请求实体过大，超出服务器的处理能力。   
414   （请求的 URI 过长） 请求的 URI（通常为网址）过长，服务器无法处理。   
415   （不支持的媒体类型） 请求的格式不受请求页面的支持。   
416   （请求范围不符合要求） 如果页面无法提供请求的范围，则服务器会返回此状态代码。   
417   （未满足期望值） 服务器未满足"期望"请求标头字段的要求。

**5开头（服务器错误）这些状态代码表示服务器在尝试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   （服务器内部错误）  服务器遇到错误，无法完成请求。   
501   （尚未实施） 服务器不具备完成请求的功能。 例如，服务器无法识别请求方法时可能会返回此代码。   
502   （错误网关） 服务器作为网关或代理，从上游服务器收到无效响应。   
503   （服务不可用） 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。   
504   （网关超时）  服务器作为网关或代理，但是没有及时从上游服务器收到请求。   
505   （HTTP 版本不受支持） 服务器不支持请求中所用的 HTTP 协议版本。

**24. 如何进行网站性能优化**

1. 从用户角度而言，优化能够让页面加载得更快、对用户的操作响应得更及时，能够给用户提供更为友好的体验。  
2. 从服务商角度而言，优化能够减少页面请求数、或者减小请求所占带宽，能够节省可观的资源。  
　　总之，恰当的优化不仅能够改善站点的用户体验并且能够节省相当的资源利用。  
　　前端优化的途径有很多，按粒度大致可以分为两类，第一类是页面级别的优化，例如 HTTP请求数、脚本的无阻塞加载、内联脚本的位置优化等 ;第二类则是代码级别的优化，例如 Javascript中的DOM 操作优化、CSS选择符优化、图片优化以及 HTML结构优化等等。另外，本着提高投入产出比的目的，后文提到的各种优化策略大致按照投入产出比从大到小的顺序排列。  
　　一、页面级优化  
1. JavaScript 压缩和模块打包  
2. 按需加载资源  
3. 在使用 DOM 操作库时用上 array-ids  
4. 缓存  
5. 启用 HTTP/2  
6. 应用性能分析  
7. 使用负载均衡方案  
8. 为了更快的启动时间考虑一下同构  
9. 使用索引加速数据库查询  
10. 使用更快的转译方案  
11. 避免或最小化 JavaScript 和 CSS 的使用而阻塞渲染  
12. 用于未来的一个建议：使用 service workers + 流  
13. 图片编码优化

**25. react和vue有哪些不同，说说你对这两个框架的看法**

相同点

· 都支持服务器端渲染

· 都有Virtual DOM,组件化开发,通过props参数进行父子组件数据的传递,都实现webComponent规范

· 数据驱动视图

· 都有支持native的方案,React的React native,Vue的weex

不同点

· React严格上只针对MVC的view层,Vue则是MVVM模式

· virtual DOM不一样,vue会跟踪每一个组件的依赖关系,不需要重新渲染整个组件树.而对于React而言,每当应用的状态被改变时,全部组件都会重新渲染,所以react中会需要shouldComponentUpdate这个生命周期函数方法来进行控制

· 组件写法不一样, React推荐的做法是 JSX + inline style, 也就是把HTML和CSS全都写进JavaScript了,即'all in js'; Vue推荐的做法是webpack+vue-loader的单文件组件格式,即html,css,jd写在同一个文件;

· 数据绑定: vue实现了数据的双向绑定,react数据流动是单向的

· state对象在react应用中不可变的,需要使用setState方法更新状态;在vue中,state对象不是必须的,数据由data属性在vue对象中管理

**26.什么是mvvm mvc是什么区别 原理**

一、MVC（Model-View-Controller）

MVC是比较直观的架构模式，用户操作->View（负责接收用户的输入操作）->Controller（业务逻辑处理）->Model（数据持久化）->View（将结果反馈给View）。

MVC使用非常广泛，比如JavaEE中的SSH框架

三、MVVM（Model-View-ViewModel）

如果说MVP是对MVC的进一步改进，那么MVVM则是思想的完全变革。它是将“数据模型数据双向绑定”的思想作为核心，因此在View和Model之间没有联系，通过ViewModel进行交互，而且Model和ViewModel之间的交互是双向的，因此视图的数据的变化会同时修改数据源，而数据源数据的变化也会立即反应view。

**27.px和em的区别**

px表示像素 (计算机屏幕上的一个点：1px = 1/96in)，是绝对单位，不会因为其他元素的尺寸变化而变化；

·

·

em表示相对于父元素的字体大小。em是相对单位 ，没有一个固定的度量值，而是由其他元素尺寸来决定的相对值。

**28.优雅降级和渐进增强**

**渐进增强（Progressive Enhancement）**：一开始就针对低版本浏览器进行构建页面，完成基本的功能，然后再针对高级浏览器进行效果、交互、追加功能达到更好的体验。

**优雅降级（Graceful Degradation）**：一开始就构建站点的完整功能，然后针对浏览器测试和修复。比如一开始使用 CSS3 的特性构建了一个应用，然后逐步针对各大浏览器进行 hack 使其可以在低版本浏览器上正常浏览。

其实渐进增强和优雅降级并非什么新概念，只是旧的概念换了一个新的说法。在传统软件开发中，经常会提到**向上兼容**和**向下兼容**的概念。渐进增强相当于向上兼容，而优雅降级相当于向下兼容

**29.eval()的作用**

把字符串参数解析成JS代码并运行，并返回执行的结果；

eval("2+3");//执行加运算，并返回运算值。

eval("varage=10");//声明一个age变量

**eval的作用域**

functiona(){

1.  eval("var x=1"); //等效于 var x=1;

2.  console.log(x); //输出1

3. }

4. a();

5. console.log(x);//错误 x没有定

**30. JS哪些操作会造成内存泄露**

1）意外的全局变量引起的内存泄露

function leak(){

  leak="xxx";//leak成为一个全局变量，不会被回收

}

2）闭包引起的内存泄露

3）3）没有清理的DOM元素引用

4）被遗忘的定时器或者回调 5）子元素存在引起的内存泄露

**31. 浏览器缓存有哪些，通常缓存有哪几种**

一、http缓存

二、websql

cookie

localstorage

sessionstorage

flash缓存

**32:bootstrap响应式实现的原理**

百分比布局+媒体查询

**33.关于JS事件冒泡与JS事件代理（事件委托）**

 事件作为DOM操作重要的一环，需要大家好好理解和运用，今天特意看了一下事件冒泡和事件代理的相关资料，感触颇深，也深感自己的无知不知道多浪费了多少内存，废话不多说进入正题：

**1.事件冒泡：**

      通俗易懂的来讲，就是当一个子元素的事件被触发的时候（如onclick事件），该事件会从事件源（被点击的子元素）开始逐级向上传播，触发父级元素的点击事件。

**2.事件委托**

    事件委托，首先按字面的意思就能看你出来，是将事件交由别人来执行，再联想到上面讲的事件冒泡，是不是想到了？对，就是将子元素的事件通过冒泡的形式交由父元素来执行。下面经过详细的例子来说明事件委托：

有可能在开发的时候会遇到这种情况：如导航每一个栏目都要加一个事件，你可能会通过遍历来给每个栏目添加事件：

事件委托是怎

1. **var** ul = document.getElementById('parentUl');

2.     ul.onclick=**function** (event) {

3.       **var** e = event||window.event,

4.               source = e.target || e.srcElement;//target表示在事件冒泡中触发事件的源元素，在IE中是srcElement

5.         **if**(source.nodeName.toLowerCase() == "li"){   //判断只有li触发的才会输出内容

6.             alert(source.innerHTML);

7.         }

8.         stopPropagation(e);                           //阻止继续冒泡

9.     };

10.     **function** addElement() {

11.         **var** li = document.createElement('li');

12.         li.innerHTML="我是新孩子";

13.         ul.appendChild(li);

14.     }

**34. CSS样式覆盖规则**

规则一：由于继承而发生样式冲突时，最近祖先获胜。

规则二：继承的样式和直接指定的样式冲突时，直接指定的样式获胜

规则三：直接指定的样式发生冲突时，样式权值高者获胜。

样式的权值取决于样式的选择器，权值定义如下表。

|  |  |
| --- | --- |
| CSS选择器 | 权值 |
| 标签选择器 | 1 |
| 类选择器 | 10 |
| ID选择器 | 100 |
| 内联样式 | 1000 |
| 伪元素(:first-child等) | 1 |
| 伪类(:link等) | 10 |

可以看到，内联样式的权值>>ID选择器>>类选择器>>标签选择器，除此以外，后代选择器的权值为每项权值之和，比如”#nav .current a”的权值为100 + 10 + 1 = 111。

规则四：样式权值相同时，后者获胜。

规则五：!important的样式属性不被覆盖。

!important可以看做是万不得已的时候，打破上述四个规则的”金手指”。如果你一定要采用某个样式属性，而不让它被覆盖的，可以在属性值后加上!important，以规则四的例子为例，”.byline a {color:red !important;}”可以强行使链接显示红色。大多数情况下都可以通过其他方式来控制样式的覆盖，不能滥用!important。

**35. 介绍一下box-sizing属性**

**兼容问题**   
首先，box-sizing属性在FireFox中存在兼容问题，所以需要使用-moz-box-sizing做一下兼容。

**属性值**

· box-sizing：content-box

· box-sizing：border-box

· box-sizing：inherit

**content-box**

· 这是box-sizing的默认属性值

· 是CSS2.1中规定的宽度高度的显示行为

· 在CSS中定义的宽度和高度就对应到元素的内容框

· 在CSS中定义的宽度和高度之外绘制元素的内边距和边框

**border-box**

· 在CSS中微元素设定的宽度和高度就决定了元素的边框盒

· 即为元素在设置内边距和边框是在已经设定好的宽度和高度之内进行绘制

· CSS中设定的宽度和高度减去边框和内间距才能得到元素内容所占的实际宽度和高度

**36. css选择符有哪些？优先级算法如何计算？（常见）**

**37. 请简要描述margin重合问题，及解决方式**

1.同向margin的重叠：  
1图片的margin-top与3图片的margin-top发生重叠，2图片的[margin-bottom](https://www.baidu.com/s?wd=margin-bottom&tn=SE_PcZhidaonwhc_ngpagmjz&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1d9m1P9mvN9uA7WPjn4rj7h0ZwV5Hcvrjm3rH6sPfKWUMw85NP7gdKWyh9GuA7EUMICmdq8uLK9uvdJX6K1TL0qnfK1TL0z5HD0IgF_5y9YIZ0lQzqlpA-bmyt8mh7GuZR8mvqVQL7dugPYpyq8Q1m3rH0vPjDdn1fYPHRYPHRvP0)与3图片的[margin-bottom](https://www.baidu.com/s?wd=margin-bottom&tn=SE_PcZhidaonwhc_ngpagmjz&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1d9m1P9mvN9uA7WPjn4rj7h0ZwV5Hcvrjm3rH6sPfKWUMw85NP7gdKWyh9GuA7EUMICmdq8uLK9uvdJX6K1TL0qnfK1TL0z5HD0IgF_5y9YIZ0lQzqlpA-bmyt8mh7GuZR8mvqVQL7dugPYpyq8Q1m3rH0vPjDdn1fYPHRYPHRvP0)发生重叠。这时候重叠之后的margin值由发生重叠两片的最大值决定；如果其中一个出现负值，则由最大的正边距减去绝对值最大的负边距，如果没有最大正边距，则由0减去绝对值最大的负边距。  
解决同向重叠的方法：  
（1）在最外层的div中加入overflow:hidden;zoom:1  
（2）在最外层加入padding:1px;属性  
（3）在最外层加入：border:1px solid #cacbcc;  
2.异向重叠问题：  
1图片的[margin-bottom](https://www.baidu.com/s?wd=margin-bottom&tn=SE_PcZhidaonwhc_ngpagmjz&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1d9m1P9mvN9uA7WPjn4rj7h0ZwV5Hcvrjm3rH6sPfKWUMw85NP7gdKWyh9GuA7EUMICmdq8uLK9uvdJX6K1TL0qnfK1TL0z5HD0IgF_5y9YIZ0lQzqlpA-bmyt8mh7GuZR8mvqVQL7dugPYpyq8Q1m3rH0vPjDdn1fYPHRYPHRvP0)与2图片的margin-top发生重叠，这时候重叠之后的margin值由发生重叠两图片的最大值的决定的。  
解决异向重叠问题：  
float:left（只能解决IE6浏览器中的异向重叠问题，可以解决IE8以上、chorme、firefox、opera下的同向重叠问题）

**38：position的值，relative\absolute\fixed分别相对于进行谁定位，有什么区别，什么时候用？**

**39.解释下CSS sprites，以及你要如何在页面或网站中使用它。**

CSS Sprites其实就是把网页中一些背景图片整合到一张图片文件中，再利用CSS的“background-image”，“background-repeat”，“background-position”的组合进行背景定位，background-position可以用数字能精确的定位出背景图片的位置

**40.什么是闭包，如何使用它，为什么要使用它？**

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

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

使用闭包的注意点：

· 由于闭包会使得函数中的变量都被保存在内存中，内存消耗很大，所以不能滥用闭包，否则会造成网页的性能问题，在IE中可能导致内存泄露。解决方法是，在退出函数之前，将不使用的局部变量全部删除。

· 闭包会在父函数外部，改变父函数内部变量的值。所以，如果你把父函数当作对象（object）使用，把闭包当作它的公用方法（Public Method），把内部变量当作它的私有属性（private value），这时一定要小心，不要随便改变父函数内部变量的值。

**41.请解释JSONP的工作原理，以及它为什么不是真正的AJAX。**

JSONP (JSON with Padding)是一个简单高效的跨域方式，HTML中的script标签可以加载并执行其他域的javascript，于是我们可以通过script标记来动态加载其他域的资源。例如我要从域A的页面pageA加载域B的数据，那么在域B的页面pageB中我以JavaScript的形式声明pageA需要的数据，然后在 pageA中用script标签把pageB加载进来，那么pageB中的脚本就会得以执行。JSONP在此基础上加入了回调函数，pageB加载完之后会执行pageA中定义的函数，所需要的数据会以参数的形式传递给该函数。JSONP易于实现，但是也会存在一些安全隐患，如果第三方的脚本随意地执行，那么它就可以篡改页面内容，截获敏感数据。但是在受信任的双方传递数据，JSONP是非常合适的选择。

AJAX是不跨域的，而JSONP是一个是跨域的，还有就是二者接收参数形式不一样！

**42.请解释一下JavaScript的同源策略。**

在客户端编程语言中，如javascript和 ActionScript，同源策略是一个很重要的安全理念，它在保证数据的安全性方面有着重要的意义。同源策略规定跨域之间的脚本是隔离的，一个域的脚本不能访问和操作另外一个域的绝大部分属性和方法。那么什么叫相同域，什么叫不同的域呢？当两个域具有相同的协议, 相同的端口，相同的host，那么我们就可以认为它们是相同的域。同源策略还应该对一些特殊情况做处理，比如限制file协议下脚本的访问权限。本地的HTML文件在浏览器中是通过file协议打开的，如果脚本能通过file协议访问到硬盘上其它任意文件，就会出现安全隐患，目前IE8还有这样的隐患。

**43.怎样添加、移除、移动、复制、创建和查找节点？**

 1）创建新节点

createDocumentFragment() //创建一个DOM片段  
createElement() //创建一个具体的元素  
createTextNode() //创建一个文本节点

2）添加、移除、替换、插入  
appendChild() //添加  
removeChild() //移除  
replaceChild() //替换  
insertBefore() //插入

3）查找  
getElementsByTagName() //通过标签名称  
getElementsByName() //通过元素的Name属性的值  
getElementById() //通过元素Id，唯一性

**44.谈谈垃圾回收机制方式及内存管理**

回收机制方式

1、定义和用法：垃圾回收机制(GC:Garbage Collection),执行环境负责管理代码执行过程中使用的内存。

2、原理：垃圾收集器会定期（周期性）找出那些不在继续使用的变量，然后释放其内存。但是这个过程不是实时的，因为其开销比较大，所以垃圾回收器会按照固定的时间间隔周期性的执行。

3、实例如下：

function fn1() {

    var obj = {name: 'hanzichi', age: 10};

}

function fn2() {

    var obj = {name:'hanzichi', age: 10};

   return obj;

}var a = fn1();var b = fn2();

fn1中定义的obj为局部变量，而当调用结束后，出了fn1的环境，那么该块内存会被js引擎中的垃圾回收器自动释放；在fn2被调用的过程中，返回的对象被全局变量b所指向，所以该块内存并不会被释放。

 4、垃圾回收策略：标记清除(较为常用)和引用计数。

**标记清除：**

　　定义和用法：当变量进入环境时，将变量标记"进入环境"，当变量离开环境时，标记为："离开环境"。某一个时刻，垃圾回收器会过滤掉环境中的变量，以及被环境变量引用的变量，剩下的就是被视为准备回收的变量。

　　到目前为止，IE、Firefox、Opera、Chrome、Safari的js实现使用的都是标记清除的垃圾回收策略或类似的策略，只不过垃圾收集的时间间隔互不相同。

**引用计数：**

　　定义和用法：引用计数是跟踪记录每个值被引用的次数。

　　基本原理：就是变量的引用次数，被引用一次则加1，当这个引用计数为0时，被视为准备回收的对象。

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

**(1)、bind 【jQuery 1.3之前】**

定义和用法：主要用于给选择到的元素上绑定特定事件类型的监听函数；

语法：bind(type,[data],function(eventObject))；

特点：

(1)、适用于页面元素静态绑定。只能给调用它的时候已经存在的元素绑定事件，不能给未来新增的元素绑定事件。

(2)、当页面加载完的时候，你才可以进行bind()，所以可能产生效率问题。

实例如下：$( "#members li a" ).bind( "click", function( e ) {} );

**(2)、live 【jQuery 1.3之后】**

定义和用法：主要用于给选择到的元素上绑定特定事件类型的监听函数；

语法：live(type, [data], fn);

特点：

(1)、live方法并没有将监听器绑定到自己(this)身上，而是绑定到了this.context上了。

(2)、live正是利用了事件委托机制来完成事件的监听处理，把节点的处理委托给了document，新添加的元素不必再绑定一次监听器。

(3)、使用live（）方法但却只能放在直接选择的元素后面，不能在层级比较深，连缀的DOM遍历方法后面使用，即$(“ul”").live...可以，但$("body").find("ul").live...不行；

实例如下：$( document ).on( "click", "#members li a", function( e ) {} );

**(3)、delegate 【jQuery 1.4.2中引入】**

定义和用法：将监听事件绑定在就近的父级元素上

语法：delegate(selector,type,[data],fn)

特点：

(1)、选择就近的父级元素，因为事件可以更快的冒泡上去，能够在第一时间进行处理。

(2)、更精确的小范围使用事件代理，性能优于.live()。可以用在动态添加的元素上。

实例如下：

$("#info\_table").delegate("td","click",function(){/\*显示更多信息\*/});

$("table").find("#info").delegate("td","click",function(){/\*显示更多信息\*/});

**(4)、on 【1.7版本整合了之前的三种方式的新事件绑定机制】**

定义和用法：将监听事件绑定到指定元素上。

语法：on(type,[selector],[data],fn)

实例如下：$("#info\_table").on("click","td",function(){/\*显示更多信息\*/});参数的位置写法与delegate不一样。

说明：on方法是当前JQuery推荐使用的事件绑定方法，附加只运行一次就删除函数的方法是one()。

 总结：.bind(), .live(), .delegate(),.on()分别对应的相反事件为：.unbind(),.die(), .undelegate(),.off()

**46、px和em的区别**

相同点：px和em都是长度单位；

异同点：px的值是固定的，指定是多少就是多少，计算比较容易。em得值不是固定的，并且em会继承父级元素的字体大小。  
浏览器的默认字体高都是16px。所以未经调整的浏览器都符合: 1em=16px。那么12px=0.75em, 10px=0.625em。

**47、浏览器的内核分别是什么?**

IE: trident内核

Firefox：gecko内核

Safari：webkit内核

Opera：以前是presto内核，Opera现已改用Google Chrome的Blink内核

Chrome：Blink(基于webkit，Google与Opera Software共同开发)

**48、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：

a. 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

b. 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

c. 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

**49、sessionStorage 、localStorage 和 cookie 之间的区别**

 共同点：用于浏览器端存储的缓存数据

不同点：

(1)、存储内容是否发送到服务器端：当设置了Cookie后，数据会发送到服务器端，造成一定的宽带浪费；

        web storage,会将数据保存到本地，不会造成宽带浪费；

(2)、数据存储大小不同：Cookie数据不能超过4K,适用于会话标识；web storage数据存储可以达到5M;

(3)、数据存储的有效期限不同：cookie只在设置了Cookid过期时间之前一直有效，即使关闭窗口或者浏览器；

        sessionStorage,仅在关闭浏览器之前有效；localStorage,数据存储永久有效；

(4)、作用域不同：cookie和localStorage是在同源同窗口中都是共享的；sessionStorage不在不同的浏览器窗口中共享，即使是同一个页面；

**50、浏览器是如何渲染页面的？**

渲染的流程如下：

1.解析HTML文件，创建DOM树。

   自上而下，遇到任何样式（link、style）与脚本（script）都会阻塞（外部样式不阻塞后续外部脚本的加载）。

2.解析CSS。优先级：浏览器默认设置<用户设置<外部样式<内联样式<HTML中的style样式；

3.将CSS与DOM合并，构建渲染树（Render Tree）

4.布局和绘制，重绘（repaint）和重排（reflow）

**51:js的基本数据类型**

JavaScript中有五种基本数据类型，它们分别是：undefined，null，boolean，number，string。

还有一种复杂数据类型-object。

**52：事件委托**

事件委托就是利用的DOM事件的事件捕获阶段。把具体dom上发生的事件，委托给更大范围的dom去处理。好比送信员，如果每次都把信件送给每一户，非常繁琐。但是如果交给一个大范围的管理者，比如小区的传达室，那么事情会变得非常简单。事件委托就类似这种原理，我页面中有很多按钮，如果不使用事件委托，我只能在每个按钮上注册事件。非常麻烦。但如果我把事件注册在一个大范围的div（假设所有的按钮都在这个div中），那么我只要注册一次事件，就可以处理所有按钮（只要按钮包含在上述div中）事件的响应了

**53：CSS3新增了很多的属性，下面一起来分析一下新增的一些属性：**

1.CSS3边框：

· border-radius：CSS3圆角边框。在 CSS2 中添加圆角矩形需要技巧，我们必须为每个圆角使用不同的图片，在 CSS3 中，创建圆角是非常容易的，在 CSS3 中，border-radius 属性用于创建圆角。border：2px solid;

· box-shadow：CSS3边框阴影。在 CSS3 中，box-shadow 用于向方框添加阴影。box-shadow:10px 10px 5px #888888;

· border-image：CSS3边框图片。通过 CSS3 的 border-image 属性，您可以使用图片来创建边框。border-image：url(border.png) 30 30 round;

2.CSS3背景：

· background-size： 属性规定背景图片的尺寸。在 CSS3 之前，背景图片的尺寸是由图片的实际尺寸决定的。在 CSS3 中，可以规定背景图片的尺寸，这就允许我们在不同的环境中重复使用背景图片。您能够以像素或百分比规定尺寸。如果以百分比规定尺寸，那么尺寸相对于父元素的宽度和高度。

· background-origin ：属性规定背景图片的定位区域。背景图片可以放置于 content-box、padding-box 或 border-box 区域。

3.CSS3文字效果：

· text-shadow：在 CSS3 中，text-shadow 可向文本应用阴影。text-shadow:5px 5px 5px #FFFFFF;

· word-wrap :单词太长的话就可能无法超出某个区域，允许对长单词进行拆分，并换行到下一行：p{word-wrap:break-word;}

4.CSS3 2D转换：

transform：通过 CSS3 转换，我们能够对元素进行移动、缩放、转动、拉长或拉伸。

· translate()：元素从其当前位置移动，根据给定的 left（x 坐标） 和 top（y 坐标） 位置参数：transform：translate（50px,100px）;值 translate(50px,100px) 把元素从左侧移动 50 像素，从顶端移动 100 像素。

· rotate()：元素顺时针旋转给定的角度。允许负值，元素将逆时针旋转。transform:rotate(30deg);值 rotate(30deg) 把元素顺时针旋转 30 度。

· scale():元素的尺寸会增加或减少，根据给定的宽度（X 轴）和高度（Y 轴）参数：transform:scale(2,4);值 scale(2,4) 把宽度转换为原始尺寸的 2 倍，把高度转换为原始高x() 5.CSS3 3D转换：

· rotateX()：元素围绕其 X 轴以给定的度数进行旋转。transform：rotateX(120deg);

· rotateY()：元素围绕其 Y 轴以给定的度数进行旋转。transform：rotateY(120deg);

6.CSS3 过渡：当元素从一种样式变换为另一种样式时为元素添加效果。

7.CSS3动画：通过 CSS3，我们能够创建动画，这可以在许多网页中取代动画图片、Flash 动画以及 JavaScript。

8.CSS3多列：

· column-count：属性规定元素应该被分隔的列数。

· column-gap：属性规定列之间的间隔。

· column-rule ：属性设置列之间的宽度、样式和颜色规则。

9.CSS3用户界面：

· resize：属性规定是否可由用户调整元素尺寸。

· box-sizing：属性允许您以确切的方式定义适应某个区域的具体内容。

· outline-offset ：属性对轮廓进行偏移，并在超出边框边缘的位置绘制轮廓。

**54：从输入url到显示页面，都经历了什么**

第一步：客户机提出域名解析请求,并将该请求发送给本地的域名服务器。

第二步：当本地的域名服务器收到请求后,就先查询本地的缓存,如果有该纪录项,则本地的域名服务器就直接把查询的结果返回。

第三步：如果本地的缓存中没有该纪录,则本地域名服务器就直接把请求发给根域名服务器,然后根域名服务器再返回给本地域名服务器一个所查询域(根的子域)的主域名服务器的地址。  
第四步：本地服务器再向上一步返回的域名服务器发送请求,然后接受请求的服务器查询自己的缓存,如果没有该纪录,则返回相关的下级的域名服务器的地址。  
第五步：重复第四步,直到找到正确的纪录。

**第2种解释：**

一般会经历以下几个过程：

1、首先，在浏览器地址栏中输入url

2、浏览器先查看**浏览器缓存**-系统缓存-路由器缓存，如果缓存中有，会直接在屏幕中显示页面内容。若没有，则跳到第三步操作。

3、在发送http请求前，需要**域名解析**(DNS解析)(DNS（域名系统，Domain Name System）是互联网的一项核心服务，它作为可以将域名和IP地址相互映射的一个分布式数据库，能够使人更方便的访问互联网，而不用去记住IP地址。)，解析获取相应的IP地址。

4、浏览器向服务器发起**tcp连接**，与浏览器建立**tcp三次握手**。（TCP即[传输控制协议](https://www.baidu.com/s?wd=%E4%BC%A0%E8%BE%93%E6%8E%A7%E5%88%B6%E5%8D%8F%E8%AE%AE&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y3rHcvnHR1PjuBmyF-PjKb0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnWTsnHnknjm4)。TCP连接是互联网连接协议集的一种。）

5、握手成功后，浏览器向服务器发送**http请求，请求数据包**。

6、**服务器处理收到的请**求，将数据返回至浏览器

7、**浏览器收到HTTP响应**

8、读取页面内容，**浏览器渲染，**解析html源码

9、生成Dom树、解析css样式、js交互

10、客户端和服务器交互

11、ajax查询

**55：对<meta></meta>标签有什么理解**

什么是meta标签？

引自下[W3school](http://www.w3school.com.cn/html/html_head.asp)的定义说明一下。

元数据（metadata）是关于数据的信息。

标签提供关于 HTML 文档的元数据。元数据不会显示在页面上，但是对于机器是可读的。

典型的情况是，meta 元素被用于规定页面的描述、关键词、文档的作者、最后修改时间以及其他元数据。

标签始终位于 head 元素中。

元数据可用于浏览器（如何显示内容或重新加载页面），搜索引擎（关键词），或其他 web 服务。

其实对上面的概念简单总结下就是：**<meta> 标签提供关于 HTML 文档的元数据。它不会显示在页面上，但是对于机器是可读的。可用于浏览器（如何显示内容或重新加载页面），搜索引擎（关键词），或其他 web 服务。**

meta的作用

meta里的数据是供机器解读的，告诉机器该如何解析这个页面，还有一个用途是可以添加服务器发送到浏览器的http头部内容

**56：new操作符到底到了什么**

先看代码

**[javascript]** [view plain](https://blog.csdn.net/lxcao/article/details/52792466) [copy](https://blog.csdn.net/lxcao/article/details/52792466)

1. **var** Func=**function**(){

2. };

3. **var** func=**new** Func ();

new共经过了4几个阶段

1、创建一个空对象

**[javascript]** [view plain](https://blog.csdn.net/lxcao/article/details/52792466) [copy](https://blog.csdn.net/lxcao/article/details/52792466)

1. varobj=**new** Object();

2、设置原型链

**[javascript]** [view plain](https://blog.csdn.net/lxcao/article/details/52792466) [copy](https://blog.csdn.net/lxcao/article/details/52792466)

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](https://blog.csdn.net/lxcao/article/details/52792466) [copy](https://blog.csdn.net/lxcao/article/details/52792466)

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](https://blog.csdn.net/lxcao/article/details/52792466) [copy](https://blog.csdn.net/lxcao/article/details/52792466)

1. **if** (**typeof**(result) == "object"){

2.   func=result;

3. }

4. **else**{

5.     func=obj;;

6. }

**57：h5新特性**

HTML5新特性 —— 新特性  
(1)新的语义标签和属性  
(2)表单新特性  
(3)视频和音频  
(4)Canvas绘图  
(5)SVG绘图  
(6)地理定位  
(7)拖放API  
**58：vue的生命周期**

**58：请写出你对闭包的理解，并列出简单的理解**

使用闭包主要是为了设计私有的方法和变量。闭包的优点是可以避免全局变量的污染，缺点是闭包会常驻内存，会增大内存使用量，使用不当很容易造成内存泄露。

闭包有三个特性：

1.函数嵌套函数

2.函数内部可以引用外部的参数和变量

3.参数和变量不会被垃圾回收机制回收

**59：display none visibility hidden区别？**

1.display:none是彻底消失，不在文档流中占位，浏览器也不会解析该元素；visibility:hidden是视觉上消失了，可以理解为透明度为0的效果，在文档流中占位，浏览器会解析该元素；

2.使用visibility:hidden比display:none性能上要好，display:none切换显示时visibility，页面产生回流（当页面中的一部分元素需要改变规模尺寸、布局、显示隐藏等，页面重新构建，此时就是回流。所有页面第一次加载时需要产生一次回流），而visibility切换是否显示时则不会引起回流。

**60：JavaScript中如何检测一个变量是一个String类型？请写出函数实现**

typeof(obj) === "string"

typeof obj === "string"

obj.constructor === String

**61：如何理解闭包？**

1、定义和用法：当一个函数的返回值是另外一个函数，而返回的那个函数如果调用了其父函数内部的其它变量，如果返回的这个函数在外部被执行，就产生了闭包。

2、表现形式：使函数外部能够调用函数内部定义的变量。

3、实例如下：

(1)、根据作用域链的规则，底层作用域没有声明的变量，会向上一级找，找到就返回，没找到就一直找，直到window的变量，没有就返回undefined。这里明显count 是函数内部的flag2 的那个count 。

var count=10;   //全局作用域 标记为flag1function add(){

    var count=0;    //函数全局作用域 标记为flag2

    return function(){

        count+=1;   //函数的内部作用域        alert(count);

    }

}var s = add()

s();//输出1

s();//输出2

4、变量的作用域

要理解闭包，首先必须理解Javascript特殊的变量作用域。

变量的作用域分类：全局变量和局部变量。

特点：

1、函数内部可以读取函数外部的全局变量；在函数外部无法读取函数内的局部变量。

2、函数内部声明变量的时候，一定要使用var命令。如果不用的话，你实际上声明了一个全局变量！

 5、使用闭包的注意点

1）滥用闭包，会造成内存泄漏：由于闭包会使得函数中的变量都被保存在内存中，内存消耗很大，所以不能滥用闭包，否则会造成网页的性能问题，在IE中可能导致内存泄露。解决方法是，在退出函数之前，将不使用的局部变量全部删除。

2）会改变父函数内部变量的值。所以，如果你把父函数当作对象（object）使用，把闭包当作它的公用方法（Public Method），把内部变量当作它的私有属性（private value），这时一定要小心，不要随便改变父函数内部变量的值。

**62：谈谈垃圾回收机制方式及内存管理**

回收机制方式

1、定义和用法：垃圾回收机制(GC:Garbage Collection),执行环境负责管理代码执行过程中使用的内存。

2、原理：垃圾收集器会定期（周期性）找出那些不在继续使用的变量，然后释放其内存。但是这个过程不是实时的，因为其开销比较大，所以垃圾回收器会按照固定的时间间隔周期性的执行。

3、实例如下：

function fn1() {

    var obj = {name: 'hanzichi', age: 10};

}

function fn2() {

    var obj = {name:'hanzichi', age: 10};

   return obj;

}var a = fn1();var b = fn2();

fn1中定义的obj为局部变量，而当调用结束后，出了fn1的环境，那么该块内存会被js引擎中的垃圾回收器自动释放；在fn2被调用的过程中，返回的对象被全局变量b所指向，所以该块内存并不会被释放。

 4、垃圾回收策略：标记清除(较为常用)和引用计数。

**标记清除：**

　　定义和用法：当变量进入环境时，将变量标记"进入环境"，当变量离开环境时，标记为："离开环境"。某一个时刻，垃圾回收器会过滤掉环境中的变量，以及被环境变量引用的变量，剩下的就是被视为准备回收的变量。

　　到目前为止，IE、Firefox、Opera、Chrome、Safari的js实现使用的都是标记清除的垃圾回收策略或类似的策略，只不过垃圾收集的时间间隔互不相同。

**引用计数：**

　　定义和用法：引用计数是跟踪记录每个值被引用的次数。

　　基本原理：就是变量的引用次数，被引用一次则加1，当这个引用计数为0时，被视为准备回收的对象。

**63：判断一个字符串中出现次数最多的字符，统计这个次数**

var str = 'asdfssaaasasasasaa';

var json = {};

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

    if(!json[str.charAt(i)]){

       json[str.charAt(i)] = 1;

    }else{

       json[str.charAt(i)]++;

    }

};var iMax = 0;var iIndex = '';for(var i in json){

    if(json[i]>iMax){

         iMax = json[i];

         iIndex = i;

    }

}        console.log('出现次数最多的是:'+iIndex+'出现'+iMax+'次');

**64、$(document).ready()方法和window.onload有什么区别？**

 (1)、window.onload方法是在网页中所有的元素(包括元素的所有关联文件)完全加载到浏览器后才执行的。

 (2)、$(document).ready() 方法可以在DOM载入就绪时就对其进行操纵，并调用执行绑定的函数。

**65、 jquery中$.get()提交和$.post()提交有区别吗？**

相同点：都是异步请求的方式来获取服务端的数据；

异同点：

1、请求方式不同：$.get() 方法使用GET方法来进行异步请求的。$.post() 方法使用POST方法来进行异步请求的。

2、参数传递方式不同：get请求会将参数跟在URL后进行传递，而POST请求则是作为HTTP消息的实体内容发送给Web服务器的，这种传递是对用户不可见的。

3、数据传输大小不同：get方式传输的数据大小不能超过2KB 而POST要大的多

4、安全问题： GET 方式请求的数据会被浏览器缓存起来，因此有安全问题。

**66、jQuery的事件委托方法bind 、live、delegate、on之间有什么区别？（常见）**

**(1)、bind 【jQuery 1.3之前】**

定义和用法：主要用于给选择到的元素上绑定特定事件类型的监听函数；

语法：bind(type,[data],function(eventObject))；

特点：

(1)、适用于页面元素静态绑定。只能给调用它的时候已经存在的元素绑定事件，不能给未来新增的元素绑定事件。

(2)、当页面加载完的时候，你才可以进行bind()，所以可能产生效率问题。

实例如下：$( "#members li a" ).bind( "click", function( e ) {} );

**(2)、live 【jQuery 1.3之后】**

定义和用法：主要用于给选择到的元素上绑定特定事件类型的监听函数；

语法：live(type, [data], fn);

特点：

(1)、live方法并没有将监听器绑定到自己(this)身上，而是绑定到了this.context上了。

(2)、live正是利用了事件委托机制来完成事件的监听处理，把节点的处理委托给了document，新添加的元素不必再绑定一次监听器。

(3)、使用live（）方法但却只能放在直接选择的元素后面，不能在层级比较深，连缀的DOM遍历方法后面使用，即$(“ul”").live...可以，但$("body").find("ul").live...不行；

实例如下：$( document ).on( "click", "#members li a", function( e ) {} );

**(3)、delegate 【jQuery 1.4.2中引入】**

定义和用法：将监听事件绑定在就近的父级元素上

语法：delegate(selector,type,[data],fn)

特点：

(1)、选择就近的父级元素，因为事件可以更快的冒泡上去，能够在第一时间进行处理。

(2)、更精确的小范围使用事件代理，性能优于.live()。可以用在动态添加的元素上。

实例如下：

$("#info\_table").delegate("td","click",function(){/\*显示更多信息\*/});

$("table").find("#info").delegate("td","click",function(){/\*显示更多信息\*/});

**(4)、on 【1.7版本整合了之前的三种方式的新事件绑定机制】**

定义和用法：将监听事件绑定到指定元素上。

语法：on(type,[selector],[data],fn)

实例如下：$("#info\_table").on("click","td",function(){/\*显示更多信息\*/});参数的位置写法与delegate不一样。

说明：on方法是当前JQuery推荐使用的事件绑定方法，附加只运行一次就删除函数的方法是one()。

 总结：.bind(), .live(), .delegate(),.on()分别对应的相反事件为：.unbind(),.die(), .undelegate(),.off()

**67、px和em的区别（常见）**

相同点：px和em都是长度单位；

异同点：px的值是固定的，指定是多少就是多少，计算比较容易。em得值不是固定的，并且em会继承父级元素的字体大小。  
浏览器的默认字体高都是16px。所以未经调整的浏览器都符合: 1em=16px。那么12px=0.75em, 10px=0.625em。

**68、浏览器的内核分别是什么?**

IE: trident内核

Firefox：gecko内核

Safari：webkit内核

Opera：以前是presto内核，Opera现已改用Google Chrome的Blink内核

Chrome：Blink(基于webkit，Google与Opera Software共同开发)

**69、什么叫优雅降级和渐进增强？（常见）**

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：

a. 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

b. 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

c. 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

**70、sessionStorage 、localStorage 和 cookie 之间的区别（常见）**

 共同点：用于浏览器端存储的缓存数据

不同点：

(1)、存储内容是否发送到服务器端：当设置了Cookie后，数据会发送到服务器端，造成一定的宽带浪费；

        web storage,会将数据保存到本地，不会造成宽带浪费；

(2)、数据存储大小不同：Cookie数据不能超过4K,适用于会话标识；web storage数据存储可以达到5M;

(3)、数据存储的有效期限不同：cookie只在设置了Cookid过期时间之前一直有效，即使关闭窗口或者浏览器；

        sessionStorage,仅在关闭浏览器之前有效；localStorage,数据存储永久有效；

(4)、作用域不同：cookie和localStorage是在同源同窗口中都是共享的；sessionStorage不在不同的浏览器窗口中共享，即使是同一个页面；