1、sessionStorage 、localStorage 和 cookie 之间的区别  
答：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

2、animation与transitions的区别  
Transition作用是指定了某一个属性（如width、left、transform等）在两个值之间如何过渡，他包括transition-property、transition-duration、transition-timing-function、transition-delay等。  
Animation也是通过指定某一个属性（如width、left、transform等）在两个值之间如何过渡来实现动画的，与Transition不同的是，Animation可以通过keyframe显式控制当前帧的属性值，而Transition只能隐式来进行（不能指定每帧的属性值），所以相对而言Animation的功能更加灵活。另外一个区别是Animation通过模拟属性值改变来实现动画，动画结束之后元素的属性没有变化；而Transition确实改变了元素的属性值，动画结束之后元素的属性发生了变化；这一点，这在实际应用中会产生很大的区别，也决定了二者各有春秋。

3、pc端不同浏览器兼容性问题及解决方案  
首先引入初始化css,保证消除不同浏览器的细微差别。  
具体内容：  
\*{margin:0;padding:0;}  
body{font-size:12px;line-height:25px;font-family:Arial,Helvetica,"Microsoft YaHei","SimSun",sans-serif;color:#606060;background:#f6f6f6;padding:0;margin:0;position:relative;text-align:justify;text-justify:newspaper;}  
img{border:none;}  
a{outline:none;}  
a:link{text-decoration:none;color:#606060;}  
a:visited{text-decoration:none;color:#606060;}  
a:hover{text-decoration:underline;color:#ff5600;}  
input{margin:0;padding:0;outline:none;background:#fff;font-size:12px;color:#606060;border:1px solid #ddd;}  
input[type='submit'], input[type='reset'],input[type='button'],button{cursor:pointer;border:none;-webkit-appearance:none;}  
textarea{outline:none;resize:none;font-size:12px;border:1px solid #ddd;}  
ul, ol, li{list-style-type:none;}  
.clr{display:block;clear:both;}  
.clearfix{display:block;content:"";clear:both;overflow:hidden;}  
CSS选择器 nth-of-type不使用。  
针对IE低版本浏览器及其他  
css hack  
1.IE6：\_  
2.IE6/7：\*  
3.IE7/Firefox：!important  
4.IE7：\*+  
5.IE6/7/8：\9  
6.IE8：\0  
条件hack  
1.IE7以下版本：<!--[if lt IE 7]><html class="no-js lt-ie9 lt-ie8 lt-ie7"><![endif]-->  
2.IE7：<!--[if IE 7]><html class="no-js lt-ie9 lt-ie8"><![endif]-->  
3.IE8：<!--[if IE 8]> <html class="no-js lt-ie9"><![endif]-->  
4.IE8以上：<!--[if gt IE 8]><!--><html class="no-js"><!--<![endif]-->  
360浏览器：  
360浏览器使用兼容模式打开  
在head里加上：  
 <meta name='renderer'content='webkit|ie-comp|ie-stand' />  
 <meta http-equiv="X-UA-Compatible" content="IE=IE7，chrome=1">  
IE浮动 margin产生的双倍距离  
#box {  
float:left;  
width:100px;  
margin:0 0 0 100px; //这种情况之下IE会产生200px的距离  
display:inline; //使浮动忽略  
}等。

4、link 和 @important 的区别

出处：link属于XHTML标签，而@import则是CSS提供的一种引入样式的方式；

    @import只能加载CSS，link标签除了可以加载CSS，还可以定义rel链接属性等；

兼容性：@import是css2提出的，只兼容到IE5以上，link标签不存在兼容问题；

用法： 样式表中可以使用@import再次引入其他的样式表，而使用link则不行；

渲染：link标签是跟DOM同时加载的，@import引用的样式则是等页面全部被下载完再被加载，网速慢的时候页面刚开始没有样式，用户体验不好；

推荐：综上所述，引入link标签好一点

5、解析HTML结构。

加载外部脚本和样式表文件。

解析并执行脚本代码。

构造HTML DOM模型。

加载图片等外部文件。

页面加载完毕。

6、DOM加载顺序

解析HTML结构。

加载外部脚本和样式表文件。

解析并执行脚本代码。

构造HTML DOM模型。

加载图片等外部文件。

页面加载完毕。

7、JQuery中$.ajax()方法参数详解？  
url: 要求为String类型的参数，（默认为当前页地址）发送请求的地址。  
type: 要求为String类型的参数，请求方式（post或get还有（push，options））

datatype：“json”默认为get。注意其他http请求方法，例如put和delete也可以使用，但仅部分浏览器支持。  
timeout: 要求为Number类型的参数，设置请求超时时间（毫秒）。此设置将覆盖$.ajaxSetup()方法的全局设置。  
async：要求为Boolean类型的参数，默认设置为true，所有请求均为异步请求。  
如果需要发送同步请求，请将此选项设置为false。注意，同步请求将锁住浏览器，用户其他操作必须等待请求完成才可以执行。  
cache：要求为Boolean类型的参数，默认为true（当dataType为script时，默认为false）。设置为false将不会从浏览器缓存中加载请求信息。  
data: 要求为Object或String类型的参数，发送到服务器的数据。如果已经不是字符串，将自动转换为字符串格式。get请求中将附加在url后。防止这种自动转换，可以查看processData选项。对象必须为key/value格式，例如{foo1:"bar1",foo2:"bar2"}转换为&foo1=bar1&foo2=bar2。如果是数组，JQuery将自动为不同值对应同一个名称。例如{foo:["bar1","bar2"]}转换为&foo=bar1&foo=bar2。  
dataType: 要求为String类型的参数，预期服务器返回的数据类型。如果不指定，JQuery将自动根据http包mime信息返回responseXML或responseText，并作为回调函数参数传递。可用的类型如下：  
          xml：返回XML文档，可用JQuery处理。  
          html：返回纯文本HTML信息；包含的script标签会在插入DOM时执行。  
          script：返回纯文本JavaScript代码。不会自动缓存结果。除非设置了cache参数。注意在远程请求时（不在同一个域下），所有post请求都将转为get请求。  
          json：返回JSON数据。  
          jsonp：JSONP格式。使用SONP形式调用函数时，例如myurl?callback=?，JQuery将自动替换后一个“?”为正确的函数名，以执行回调函数。  
          text：返回纯文本字符串。  
beforeSend：要求为Function类型的参数，发送请求前可以修改XMLHttpRequest对象的函数，例如添加自定义HTTP头。在beforeSend中如果返回false可以取消本次ajax请求。XMLHttpRequest对象是惟一的参数。function(XMLHttpRequest){  
   this;   //调用本次ajax请求时传递的options参数  
            }  
complete：要求为Function类型的参数，请求完成后调用的回调函数（请求成功或失败时均调用）。  
参数：XMLHttpRequest对象和一个描述成功请求类型的字符串。  
function(XMLHttpRequest, textStatus){  
             this;//调用本次ajax请求时传递的options参数  
          }  
success：要求为Function类型的参数，请求成功后调用的回调函数，有两个参数。  
         (1)由服务器返回，并根据dataType参数进行处理后的数据。  
         (2)描述状态的字符串。  
         function(data, textStatus){  
            //data可能是xmlDoc、jsonObj、html、text等等  
            this;  //调用本次ajax请求时传递的options参数  
error：要求为Function类型的参数，请求失败时被调用的函数。该函数有3个参数，即XMLHttpRequest对象、错  
       误信息、捕获的错误对象(可选)。  
       ajax事件函数如下：  
       function(XMLHttpRequest, textStatus, errorThrown){  
          //通常情况下textStatus和errorThrown只有其中一个包含信息  
          this;   //调用本次ajax请求时传递的options参数  
       }  
contentType：要求为String类型的参数，当发送信息至服务器时，内容编码类型默认为"application/x-www-form-urlencoded"。该默认值适合大多数应用场合。  
dataFilter：要求为Function类型的参数，给Ajax返回的原始数据进行预处理的函数。提供data和type两个参数。data是Ajax返回的原始数据，type是调用jQuery.ajax时提供的dataType参数。函数返回的值将由jQuery进一步处理。  
            function(data, type){  
                //返回处理后的数据  
                return data;  
            }  
global：要求为Boolean类型的参数，默认为true。表示是否触发全局ajax事件。设置为false将不会触发全局ajax事件，ajaxStart或ajaxStop可用于控制各种ajax事件。  
ifModified：要求为Boolean类型的参数，默认为false。仅在服务器数据改变时获取新数据。服务器数据改变判断的依据是Last-Modified头信息。默认值是false，即忽略头信息。  
jsonp：要求为String类型的参数，在一个jsonp请求中重写回调函数的名字。该值用来替代在"callback=?"这种GET或POST请求中URL参数里的"callback"部分，例如{jsonp:'onJsonPLoad'}会导致将"onJsonPLoad=?"传给服务器。  
username：要求为String类型的参数，用于响应HTTP访问认证请求的用户名。  
password：要求为String类型的参数，用于响应HTTP访问认证请求的密码。  
processData：要求为Boolean类型的参数，默认为true。默认情况下，发送的数据将被转换为对象（从技术角度来讲并非字符串）以配合默认内容类型"application/x-www-form-urlencoded"。如果要发送DOM树信息或者其他不希望转换的信息，请设置为false。  
scriptCharset：要求为String类型的参数，只有当请求时dataType为"jsonp"或者"script"，并且type是GET时才会用于强制修改字符集(charset)。通常在本地和远程的内容编码不同时使用。

8、px和em，rem的区别  
答：px：绝对度量单位，就是像素em：相对度量单位，但是计算起来实在费劲。  
em指的是相对于当前对象内文本的字体尺寸。如当前对行内文本的字体尺寸未被人为设置，则相对于浏览器的默认字体尺寸。换句话说，如果当前div的字体大小是12像素，那么1em就是12像素，如果div字体大小是24像素，1em就是24像素。  
rem，也是和字体相关的相对度量单位，但是，rem是相对于根元素的字体大小进行设置的，如果html元素中的字体大小设置为24px，那么针对任意html内的元素设置1rem，均表示的是24px，大大节省了我们计算字体大小的时间。

**9, Canvas 和 SVG 的区别：**

**功能上：**

**canvas可以看做一个画布其绘制出来的图形为标量图，因此，可以在canvas中引入jpg或png这类格式的图片，canvas技术比较成熟，大型网络游戏统计图表，柱状图都可以用canvas制作。**

**svg，所绘制的图形为矢量图，所以其用法上受到了限制，svg中不能引入普通的图片，因为矢量图的不会失真的效果，在项目中我们会用来做一些动态的小图标。svg适合用来做地图。**

**技术上：**

**另外从技术发面来讲canvas里面绘制的图形不能被引擎抓取，如我们要让canvas里面的一个图片跟随鼠标事件：canvas.onmouseover=function(){}。而svg里面的图形可以被引擎抓取，支持事件的绑定。另外canvas中我们绘制图形通常是通过javascript来实现，svg更多的是通过标签来来实现，如在svg中绘制正矩形形就要用<rect>，这里我们不能用属性style="width:XXX;height:XXX;"来定义。我再来介绍一个svg的js库：TWO.JS。其中包含two.js和three.js前者用于绘制二维图形，后者用于绘制三维图形。TWO.JS可以支持三种格式，svg（默认）、canvas、和WEBGL。当然也可以在普通div中引入。**

　　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。  
　　　　特点：  
　　　　 　　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  
　　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。  
　　　　特点：  
　　　    　　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

10、在一个数组中找出最大值 与最小值的方法

Math.max.apply() Math.min.apply()

11、Js和jq中如何改变属性？

Js：setAttribute() Jq: attr()

 12、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？  
查找浏览器缓存  
DNS 解析、查找该域名对应的IP 地址、重定向（301）、发出第二个GET 请求  
进行HTTP 协议会话  
客户端发送报头(请求报头)  
服务器回馈报头(响应报头)  
html 文档开始下载  
文档树建立，根据标记请求所需指定MIME 类型的文件  
文件显示  
[  
浏览器这边做的工作大致分为以下几步：  
加载：根据请求的URL 进行域名解析，向服务器发起请求，接收文件（HTML 、JS 、CSS 、图象等）。  
解析：对加载到的资源（HTML 、JS 、CSS 等）进行语法解析，建议相应的内部数据结构（比如HTML 的DOM 树，JS 的（对象）属性表，CSS 的样式规则等等）  
}

13、webSocket 如何兼容低浏览器？

Adobe Flash Socket 、 ActiveX HTMLFile (IE) 、 基于 multipart 编码发送 XHR 、

14、css3中单位px,em,rem,vh,vw,vmin,vmax的区别及浏览器支持情况。  
  
px：绝对单位，页面按精确像素展示  
em：相对单位，基准点为父节点字体的大小，如果自身定义了font-size按自身来计算（浏览器默认字体是16px），整个页面内1em不是一个固定的值。  
rem：相对单位，可理解为”root em”, 相对根节点html的字体大小来计算，CSS3新加属性， chrome/firefox/IE9+支持。  
(另外需注意chrome强制最小字体为12号，即使设置成 10px 最终都会显示成 12px，当把html的font-size设置成10px,子节点rem的计算还是以12px为基准，所以网上很多文章提到的将html的font-size设为10方便计算不是那么可取)  
vw：viewpoint width，视窗宽度，1vw等于视窗宽度的1%。  
vh：viewpoint height，视窗高度，1vh等于视窗高度的1%。  
vmin：vw和vh中较小的那个。  
vmax：vw和vh中较大的那个。  
  
vw, vh, vmin, vmax：IE9+局部支持，chrome/firefox/safari/opera支持，ios safari 8+支持，android browser4.4+支持，chrome for android39支持

15.为什么要用web做移动端  web前端在移动端中的优势  和android  ios比较  
   
1、ios迭代周期严重受限  
       挡在ios原生app面前最最重要的一个壁垒，就是苹果公司的审核，平均每个版本审核1周，每年只有寥寥数次的快速审核，即便是快速审核也需要3-4天左右的时间才能够发布。加之In House版的推广限制，app store已经成为了团队谈之色变的产物。让我们这种上午需求下午上线的团队几乎没有办法进行推进。  
       每一个原生app改动都要提前若干天进行部署，还需要考虑之后若干个月可能的变更，不单单是成本激增、风险更大，更重要的是：互联网时代的产品比拼如果技术成为了瓶颈，那这个产品必将寸步难行。  
2、android兼容性差  
       android虽然没有像ios那样的审核周期，但所换来的却是千差万别的兼容性问题，每一个android开发和qa手上没有5个以上的真机，都不好意思说自己接触过android。每一个android开发的内心深处，多多少少都对某些老爷机有说不出的怨念，经常会想，我为什么不去搞ios！  
3、用户使用成本高，导致拉新困难  
       这一条是针对app应用的劣势，想让一个用户下载并使用一个app，要远比让他从微信中授权使用一个网页应用难的多，导致了各种地推活动的奖品越来越好但效果越来越差事实。  
4、开发成本高  
       首先你要有一批ios开发，再加一批android开发，然后分别配置一批qa，即便单人的薪资已经有所回落，但app团队的成本永远是整个团队里最高的之一。单单是兼容老版就要花去不少人工成本。

16、H5优势：

1、迭代速度快

相比ios的迭代速度，H5简直就是即时响应，随时变更随时上线。用户可以在毫无感知的情况下获取到最新的变化。照片红包上线与下线只是瞬息之间，我甚至没有来得及ps我的玉照就已经被下线了。

2、兼容性强

H5的兼容性只体现在屏幕的大小和部分极特殊机型上，即便出现兼容性问题也仅仅是展示上的略微差异。相比android的夸张测试范围，H5可以说并不存在兼容性的问题。

3、传播途径广、用户使用成本低

H5在营销运行方面的优势太多，以至于我只能把他们归结为这一条。通过各种社交产品的病毒式扩散，用户只需简单的打开链接即可达到我们的宣传推广目的。

4、开发成本低

H5团队可以直接从pc前端团队演变过来，甚至后端开发兼职H5的例子并不是没有。总结起来就是：学得快、用人少、测试乐的呱呱叫。

17、app相比H5的优势：  
相比app的劣势，app的优势显得十分单薄。  
1、交互效果好，响应速度快  
  页游和客户端游戏的区别。目前对于追求产品极致体验但苦于H5无法满足的情况，原生app也是一种退路。  
2、机能操作必不可少  
  对于硬件机能的操作，app有其不可替代性，比如相机和touch等功能。  
3、用户黏度高  
  对于app应用来说，一旦拉新成功，只要产品对他有价值，一般不会轻易弃用。但H5，链接一关，刚才干什么就已经忘记了。

18. svn和git 的区别  
1.GIT是分布式的，SVN不是：  
这是GIT和其它非分布式的版本控制系统，例如SVN，CVS等，最核心的区别。如果你能理解这个概念，那么你就已经上手一半了。需要做一点声明，GIT并不是目前第一个或唯一的分布式版本控制系统。还有一些系统，例如Bitkeeper, Mercurial等，也是运行在分布式模式上的。但GIT在这方面做的更好，而且有更多强大的功能特征。  
GIT跟SVN一样有自己的集中式版本库或服务器。但，GIT更倾向于被使用于分布式模式，也就是每个开发人员从中心版本库/服务器上chect out代码后会在自己的机器上克隆一个自己的版本库。可以这样说，如果你被困在一个不能连接网络的地方时，就像在飞机上，地下室，电梯里等，你仍然能够提 交文件，查看历史版本记录，创建项目分支，等。对一些人来说，这好像没多大用处，但当你突然遇到没有网络的环境时，这个将解决你的大麻烦。  
同样，这种分布式的操作模式对于开源软件社区的开发来说也是个巨大的恩赐，你不必再像以前那样做出补丁包，通过email方式发送出去，你只需要创建一个分支，向项目团队发送一个推请求。这能让你的代码保持最新，而且不会在传输过程中丢失。GitHub.com   
  
就是一个这样的优秀案例。  
有些谣言传出来说subversion将来的版本也会基于分布式模式。但至少目前还看不出来。  
2.GIT把内容按元数据方式存储，而SVN是按文件：  
所有的资源控制系统都是把文件的元信息隐藏在一个类似.svn,.cvs等的文件夹里。如果你把.git目录的 体积大小跟.svn比较，你会发现它们差距很大。因为,.git目录是处于你的机器上的一个克隆版的版本库，它拥有中心版本库上所有的东西，例如标签，分 支，版本记录等。  
  
3.GIT分支和SVN的分支不同：  
分支在SVN中一点不特别，就是版本库中的另外的一个目录。如果你想知道是否合并了一个分支，你需要手工运行像这样的命令svn propget svn:mergeinfo，来确认代码是否被合并。感谢Ben同学指出这个特征。所以，经常会发生有些分支被遗漏的情况。  
然而，处理GIT的分支却是相当的简单和有趣。你可以从同一个工作目录下快速的在几个分支间切换。你很容易发现未被合并的分支，你能简单而快捷的合并这些文件。

4.GIT没有一个全局的版本号，而SVN有：  
目前为止这是跟SVN相比GIT缺少的最大的一个特征。你也知道，SVN的版本号实际是任何一个相应时间的源代 码快照。我认为它是从CVS进化到SVN的最大的一个突破。因为GIT和SVN从概念上就不同，我不知道GIT里是什么特征与之对应。如果你有任何的线 索，请在评论里奉献出来与大家共享。  
更新：有些读者指出，我们可以使用GIT的SHA-1来唯一的标识一个代码快照。这个并不能完全的代替SVN里容易阅读的数字版本号。但，用途应该是相同的。  
5.GIT的内容完整性要优于SVN：  
GIT的内容存储使用的是SHA-1哈希算法。这能确保代码内容的完整性，确保在遇到磁盘故障和网络问题时降低对版本库的破坏。

19、前后端分离的最终目的是问题提高工作效率。

传统的web开发模式，后端一定程度上依赖与前端页面的开发，而前端一定程度的依赖后端的数据接口的开发。整个开发过程中部分阶段处于串行状态。而前后端分离的核心思路就是打破这种串行，使得前端与后端能够并行开发，从而提高整体的工作效率。

很多公司都在尝试前后端分离，根据公司的情况，分离的程度也不同。个人感觉，前后端分离的火候，一定程度上依赖该公司前端团队的实力。

阶段1、页面分离

得益于 nodejs 的成功，前端可以不必安装后端开发环境，便可以直接编写后端模板，这样前端便直接提供的vm文件，而不需要后端去转页面，节约一部分后端的工作量。

这种模式下，前端开发工作量会稍微的增加一些，但不会太多，当你熟悉后端模板后，几乎不会有什么额外负担。

前端团队需要搭建一套自编译系统，将vm 自动转化为html，并刷新浏览器。

主要node包：gulp、velocity、browser-sync

阶段2、数据接口分离

数据接口分离，就是开发之前，前后端约定好一个接口，大家都基于这个接口开发。类似于app开发。

这样前端就可以与后端接口解耦了，在后端接口还没有开发完成时，也可以开发数据交互js。

主要工具： rap 或者 swagger

阶段3、展现层分离（展现层由前端维护）

这个也算是前后端分离的究极体了。在一部分公司，PHP承担着前后端分离的重要角色，有些人会有疑问，php不是后端语言么？

对，但是在这个阶段，PHP 只负责 MVC 模式下的 View 和 Controller 层，Model 层交由Java 或C#。关键一点，这里的 PHP 是由前端团队维护，如果交由后端开发去维护，便失去了前后端分离的意义。也因为是前端去维护，我们随其自然的将这里的 PHP 换成了 NodeJS。

该阶段对前端是一个考验，没有做后端开发的的同学入手比较困难，团队里至少有一个对mvc模式较为熟悉，且熟练掌握NodeJS 或者 PHP，在他的帮助下，最终每一个前端成员都能够独立的去写 view 和对应的 Controller。

当做到这一步的时候，前后端也就真正的分离了。我们直接完全是接口化开发，并且有需要的话，针对后端接口编写单元测试。

20、常见的一些兼容性问题

1 window.event IE专用

2 ie不支持ClassName取标签，非ie支持

3 获取非ie窗口的scrolltop值通过document.body.scrollTop 获取ie窗口的scrolltop值通过document.documentElement.scrollTop

4 通过childNodes获取子节点，ie忽略空白文本节点，非ie不忽略。

5 innerText在火狐中不可用

6 addEventListener 非ie支持，attachEvent IE专用

7 input标签的type属性在ie中不可修改

8 获取外部定义样式currentStyle[attr] IE专用 window.getComputedStyle(dom,false)[attr] 非IE

兼容写法：function getStyle(obj,name){

if (obj.currentStyle){

return obj.currentStyle[name]; //ie

}else {

return obj.getComputedStyle(obj,false)[name]; //ff

}

}

1. 自定义属性，obj.xxx，IE专用， 非IE使用getAttribute（）

21、AMD 和 CMD 的区别有哪些？

1. 对于依赖的模块，AMD 是提前执行，CMD 是延迟执行。

2. CMD 推崇依赖就近，AMD 推崇依赖前置。  
3. AMD 的 API 默认是一个当多个用，CMD 的 API 严格区分，推崇职责单一。

箭头函数的有点：不改变this 指向。