1. **HTTP状态码及其含义？**

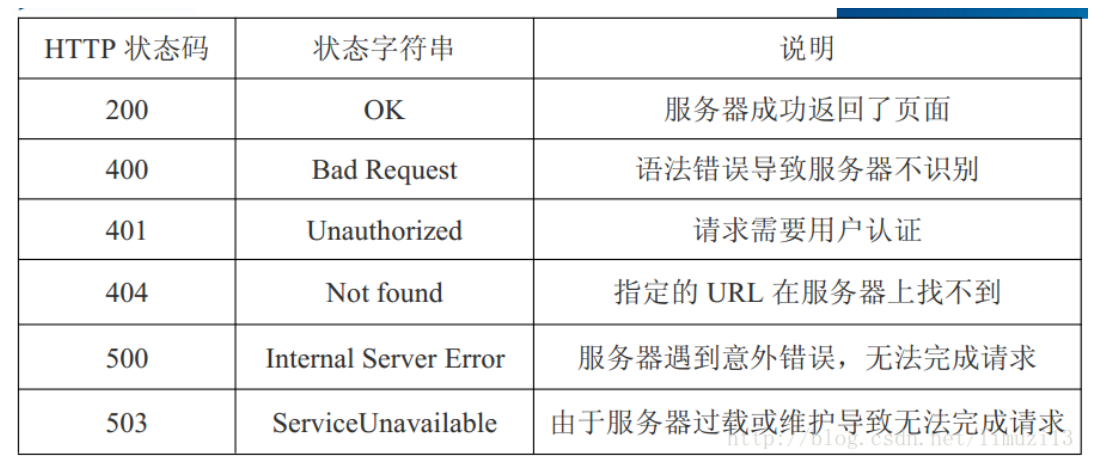
答：

100-199 : 表示成功接收请求, 要求客户端继续提交下一次请求才能完成整个处理过程

* 200-299: 表示成果接收请求并已完成整个处理过程. 常用200
* 300-399: 为完成请求, 客户需进一步细化需求: 例如: 请求的资源已经移动一个新地址, 常用302(重定向), 307和304(拿缓存)
* 400-499: 客户端的请求有错误, 包含语法错误或者不能正确执行. 常用404(请求的资源在web服务器中没有) 403(服务器拒绝访问, 权限不够)
* 500-599: 服务器端出现错误

常用:

* 200 　　 正常　　 表示一切正常, 返回的是正常请求结果
* 302/307　　临时重定向　　指出请求的文档已被临时移动到别处, 此文档的新的url在location响应头中给出
* 304　　未修改　　表示客户机缓存的版本是最新的, 客户机应该继续使用它
* 403　　禁止　　服务器理解客户端请求, 但拒绝处理它, 通常用于服务器上文件或目录的权限设置所致
* 404　　找不到　　服务器上不存在客户机所请求的资源
* 500　　服务器内部错误　　服务器端的cgi, asp, jsp等程序发生错误
* **200**：服务器响应正常。
* **304**：该资源在上次请求之后没有任何修改（这通常用于浏览器的缓存机制，使用GET请求时尤其需要注意）。
* **400**：错误请求，如语法错误
* **401**：访问资源的权限不够。
* **403**：没有权限访问资源。
* **404**：访问的资源不存在。
* **405**：访问的资源被禁止。
* **407**：访问的资源需要代理身份验证。
* **414**：请求的URL太长。
* **500**：服务器内部错误。



1. **请描述cookies、sessionStorage和localStorage的区别**

答：cookies缓存，sessionStorage会话存储，localStorage本地存储

1、cookies存储在http信息的实体中，http每次请求都会携带cookie，cookie在浏览器和服务器间来回传递，后两者仅在本地保存；

2、存储大小：cookie数据不能超过4k，后两者要大很多，可以达到5M；

3、有效期：cookie在设置的cookie有效期之前有效，即使浏览器或窗口关闭；sessionStorage仅在当前浏览器窗口关闭前有效，不可持久保存；localSorage始终有效，窗口或浏览器关闭也一直保存，可做持久数据；

4、作用域：cookie、localStorage在所有的同源窗口中都是共享的；sessionStorage只能在当前页面使用。

**cookie的弊端：**

1）Cookie数量和长度的限制。

数量：每个域的cookie总数有限。

长度：每个 cookie 长度不超过 4KB （ 4096B ），否则会被截掉。

2）潜在的安全风险 。 Cookie 可能被拦截、篡改。如果 cookie 被拦截，就有可能取得所有的 session 信息。

3）有些状态不可能保存在客户端 。

4）cookie在每次发送http请求时，都会被发送到服务器，一些不必要的信息也会被发送过去，增加额外的开销。

3、**请你解释你对盒子模型的理解，以及如何在CSS中告诉浏览器使用不同的盒子模型来渲染你的布局?**

**盒模型**：文档中的每个元素被描绘为矩形盒子，渲染引擎的目的就是判定大小，属性——比如它的颜色、背景、边框方面——及这些盒子的位置。

在网页中，一个元素占有空间的大小由几个部分构成，其中包括元素的内容（content），元素的内边距（padding），元素的外边距（margin），元素的边框（border）四个部分。

盒子模型分为两类：**标准W3C盒子模型和IE盒子模型**

　　这两者的关键差别就在于：**IE盒子模型** 的content部分的宽度和高度包含了border和pading.

我们在编写页面代码的时候应该尽量使用标准的W3C盒子模型（需要在页面中声明DOCTYPE类型）。

3.3**如何解决不同浏览器之间的兼容性问题，或者是否有遇到过兼容性问题，如何解决？**

（1）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

问题症状：随便写几个标签，不加样式控制的情况下，各自的margin 和padding差异较大。

解决方案：CSS里 \*{margin:0;padding:0;}

（2）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：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

（3）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。

（4）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;

（5）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

（6）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时不能很好的被各个浏览器兼容

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;}

（7）浏览器兼容问题七：透明度的兼容CSS设置

4、下面两个alert的结果是什么？

var foo = "Hello";

(function() {

var bar = "world";

alert(foo + bar);

})();

alert(foo + bar); //bar没有定义

考察**闭包**

第一个弹出Helloworld，第二个bar没有定义会报错。

function A(){

var a=1;

return a;

}

console.log(A()); // 1 ,这样可以访问到函数内部的局部变量

console.log(a); //undefined

**（1） function f1(){**

**var a=10;**

**function f2(){**

**console.log(a);**

**}**

**return f2;**

**}**

**//console.log(a); //a is not defined,报错**

**//为了访问方法中内部变量的值，可以通过闭包来实现**

**var f=f1();**

**f();**

**（2） function add(){**

**var count=0; //函数add中的count变量会一直保存在内存中**

**function addInner(){**

**count++;**

**console.log(count);**

**}**

**return addInner;**

**}**

**var result=add();**

**result();//1**

**result();//2**

**result();//3**

**介绍一下闭包和闭包常用场景**

（1）变量作用域

JavaScript语言的特别之处就在于：函数内部可以直接读取全局变量，但是在函数外部无法读取函数内部的局部变量。

注意点：在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，实际上声明的是一个全局变量。

（2）闭包的概念

闭包就是能够读取其他函数内部变量的函数。

在本质上，闭包是将函数内部和函数外部连接起来的桥梁。

（3）闭包的作用

最大的作用有两个：一个是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。

（4）使用闭包的注意点

①由于闭包会使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中可能导致内存泄漏。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闭包会在父函数外部，改变父函数内部变量的值。所以，如果把父函数当作对象（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。

闭包是指有权访问另一个函数作用域中的变量的函数. 创建闭包常见方式,就是在一个函数内部创建另一个函数。

应用场景：设置私有变量和方法

不适合场景：返回闭包的函数是个非常大的函数

闭包的缺点就是常驻内存，会增大内存使用量，使用不当很容易造成内存泄露。

受JavaScript链式作用域结构的影响，父级变量中无法访问到子级的变量值，为了解决这个问题，才使用闭包这个概念

5、下面代码的输出是什么？

console.log('one');

setTimeout(function() {

console.log('two');

}, 0);

console.log('three');

输出结果：one

three

two

**6、前端页面由哪三部分组成？作用是什么？**

### 答：结构层html、样式层css和行为层js、

**结构层**：

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

**样式层**：

由 CSS 负责创建。 CSS对“如何显示有关内容”的问题做出了回答。

**行为层**：

负责回答“内容应该如何对事件做出反应”这一问题。这是 Javascript 语言和 DOM主宰的领域。

**8、什么是作用域链？作用域链的特点是什么？**

答：每一个函数都有一个作用域，作用域的使用提高了程序逻辑的局部性，增强程序的可靠性，减少名字冲突。比如我们创建了一个函数，函数里面又包含了一个函数，那么现在 就有三个作用域，这样就形成了一个作用域链。

// 作用域链

// 每一个函数都有一个作用域， 如果一个函数内部又嵌套链另外一个函数，

// 嵌套的函数也会又一个作用域，这样，嵌套函数就能访问到外部函数中的变量，

// 以及全局作用域中的变量，此时就形成了一条作用域链

// （包括：自身的作用域， 以及外部函数的作用域， 以及全局作用域）

// 全局作用域： script标签包裹的内容就是一个作用域

作用域的特点就是，先在自己的变量范围中查找，如果找不到，就会沿着作用域链往上找。

**9、 一个Ajax建立的过程是怎样的，主要用到哪些状态码？**

答：ajax：在不切换页面的情况下完成异步的HTTP请求。

(1)创建XMLHttpRequest对象,也就是创建一个异步调用对象.；

(2)创建一个新的HTTP请求,并指定该HTTP请求的方法、URL及验证信息.；

(3)设置响应HTTP请求状态变化的函数.；

(4)发送HTTP请求.；

(5)获取异步调用返回的数据.；

(6)使用JavaScript和DOM实现局部刷新.；

常用状态码status

404 没找到页面(not found)  
403 禁止访问(forbidden)  
500 内部服务器出错(internal service error)  
200 一切正常(ok)  
304 没有被修改(not modified)(服务器返回304状态，表示源文件没有被修改）

19: 说说你还知道的其他状态码，状态码的存在解决了什么问题

302/307　　临时重定向

301　永久重定向

借助状态码,用户可以知道服务器端是正常处理了请求,还是出现了什么错误

10、说说你对语义化的理解

答：语义化是指用合理HTML标记以及其特有的属性去格式化文档内容。通俗地讲,语义化就是对数据和信息进行处理,使得机器可以理解。

语义化的(X)HTML文档有助于**提升**你的网站对访客的易用性。

像html5的新的标签header，footer,section等就是语义化

一方面，语义化就是让计算机能够快速的读懂内容，高效的处理信息，可以对搜索引擎更友好。

另一方面，便于与他人的协作，他人通过读代码就可以理解你网页标签的意义。

**11、介绍一下HTML5的新特性，移除的元素？**

①新的DOCTYPE声明  <!DOCTYPE[h](http://www.68idc.cn/help/server/linuxaq/2014031173216.html)tml>

②完全支持css3

③video和audio

④本地存储localStorage，长期存储数据，浏览器关闭后数据不丢失；

⑤语义化标签，例如header、footer、nav、section、article等

⑥canvas绘画

⑦新事件 如ondrag onresize

⑧表单控件，如calendar、date、time、email、url、search

⑨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。

12、前端如何进行性能优化？

答：**前端性能优化**

**雅虎14条性能优化原则**

法则1：减少HTTP请求次数

法则2：使用 CDN(Content Delivery Network, 内容分发网络 )

法则3：增加 Expires Header

法则 4：压缩页面元素

法则 5：把样式表放在头上

法则 6：把脚本文件放在底部

法则 7：避免 CSS 表达式

法则 8：把 JavaScript和 CSS 放到外部文件

法则 9：减少DNS查询次数

法则 10：最小化 JavaScript代码

法则 11：避免重定向

法则 12：删除重复的脚本文件

法则 13：配置 ETags

法则 14：缓存 Ajax

也可以这样答：

内容方面：

1.减少 HTTP 请求 (Make Fewer HTTP Requests)

2.减少 DOM 元素数量 (Reduce the Number of DOM Elements)

3.使得 Ajax 可缓存 (Make Ajax Cacheable)

针对CSS：

1.把 CSS 放到代码页上端 (Put Stylesheets at the Top)

2.从页面中剥离 JavaScript 与 CSS (Make JavaScript and CSS External)

3.精简 JavaScript 与 CSS (Minify JavaScript and CSS)

4.避免 CSS 表达式 (Avoid CSS Expressions)

针对JavaScript ：

1. 脚本放到 HTML 代码页底部 (Put Scripts at the Bottom)

2. 从页面中剥离 JavaScript 与 CSS (Make JavaScript and CSS External)

3. 精简 JavaScript 与 CSS (Minify JavaScript and CSS)

4. 移除重复脚本 (Remove Duplicate Scripts)

针对图片(Image)：

1. 使用 CSS Sprites 技巧对图片优化

2 不要在 HTML 中使用缩放图片

3 使用恰当的图片格式

13、ES6

14、**语义化的理解**

1. html语义化就是让页面的内容结构化，便于对浏览器、搜索引擎解析；
2. 在没有样式CCS情况下也以一种文档格式显示，并且是容易阅读的；
3. 搜索引擎的爬虫依赖于标记来确定上下文和各个关键字的权重，利于 SEO；
4. 使阅读源代码的人对网站更容易将网站分块，便于阅读维护理解。

14.1页面重构怎么操作？

编写 CSS、让页面结构更合理化，提升用户体验，实现良好的页面效果和提升性能。

**15、如何实现浏览器内多个标签页之间的通信？**

答：调用localstorage或cookie+setInterval()等本地存储方式。

**16、如何添加、移除、移动、复制、创建和查找节点等**

答：添加节点：appendChild()

移除节点：removeChild()

替换节点：replaceChild()

插入节点：insertBefore()

创建节点：createElement() //创建一个具体的元素

createTextNode() //创建一个文本节点

createDocumentFragment() //创建一个DOM片段

查找节点：getElementById() //通过元素的id，唯一性

getElementByName() //通过元素的Name属性的值

getElementByTagName() //通过标签名称

getElementByClassName() //通过类名

17、Doctype的作用？严格模式与 模式如何区分？它们有何意义？

答：<!DOCTYPE>声明叫做文件类型定义（DTD），位于文档中的最前面，处于<html>标签之前。告知浏览器的解析器该文件的类型是什么，用什么文档类型规范来解析这个文档。

**严格模式：**又称标准模式，是指浏览器按照 W3C 标准解析代码。

**混杂模式：**又称怪异模式或兼容模式，是指浏览器用自己的方式解析代码。

**如何区分**：浏览器解析时到底使用严格模式还是混杂模式，与网页中的 DTD 直接相关。

DOCTYPE 不存在或形式不正确会导致文档以混杂模式呈现。

**意义**：严格模式与混杂模式存在的意义与其来源密切相关，如果说只存在严格模式，那么许多旧网站必然受到影响，如果只存在混杂模式，那么会回到当时浏览器大战时的混乱，每个浏览器都有自己的解析模式。

18、行内元素有哪些？块级元素有哪些？空（void）元素有哪些？

答：CSS规范规定，每个元素都有display属性，确定该元素的类型，每个元素都有默认的display值，比如div默认display属性值为“block”，成为“块级”元素；span默认display属性值为“inline”，是“行内”元素;img默认display属性值为“inline-block”，是“内联块状”元素。    
  
常用的块状元素有：  
<div>、<p>、<h1>...<h6>、<ol>、<ul>、<dl>、<table>、<address>、<blockquote> 、<form>  
块级元素特点：  
1、每个块级元素都从新的一行开始，并且其后的元素也另起一行。（真霸道，一个块级元素独占一行）  
2、元素的高度、宽度、行高以及顶和底边距都可设置。  
3、元素宽度在不设置的情况下，是它本身父容器的100%（和父元素的宽度一致），除非设定一个宽度。  
  
常用的行内元素有：  
<a>、<span>、<br>、<i>、<em>、<strong>、<label>、<q>、<var>、<cite>、<code>  
行内元素特点：  
1、和其他元素都在一行上；  
2、元素的高度、宽度、行高及顶部和底部边距不可设置；  
3、元素的宽度就是它包含的文字或图片的宽度，不可改变。

知名的空元素： <br/> <hr/> <img/> <input/> <link/> <meta/>   
鲜为人知的是： <area> <base> <col> <command> <embed> <keygen> <param> <source> <track> <wbr>

**19、Vue如何和后台进行数据交互？**

答：①配合jQuery的ajax；②配合vue-resource组件；

具体举例如下：  
1、导入vue-resource  
<script src="js/vue.js"></script>  
<script src="js/vue-resource.js"></script>  
2、基于全局Vue对象使用http  
// 通过someUrl获取后台数据，成功后执行then的代码  
Vue.http.get('/someUrl', [options]).then(successCallback, errorCallback);  
3、在一个Vue实例内使用$http  
// $http是在vue的局部范围内的实例  
this.$http.get('/someUrl', [options]).then(successCallback, errorCallback);  
说明：  
在发送请求后，使用then方法来处理响应结果，then方法有两个参数，第一个参数是响应成功时的回调函数，第二个参数是响应失败时的回调函数。

**③使用axios**

axios({

url:"www.baidu.com?pageStart=18pageSize=3",

method:"get/post",

如果是get请求的话 用params 来传输数据或者地址栏拼接

params:{

pageStart:1,

pageSize:3

}

如果是post请求的话 用data去传输数据

data:{

}

})

20、**link和@import的区别是什么？**

答：① link是XHTML标签，除了可以引用CSS样式外，还可以定义RSS等，但@import是CSS标签，只能引用CSS样式。

②link在页面加载的同时加载，而@import是在页面内容加载完成之后加载的。

③link是XHTML标签，没有兼容问题，而@import是在CSS2.1提出来的，低版本的浏览器不支持。

④link支持使用javascript控制DOM去改变样式，@import不支持。

**21、iframe有哪些缺点？**

答：①iframe会阻塞主页面的onload事件；window 的 onload 事件需要在所有 iframe 加载完毕后(包含里面的元素)才会触发，在 Safari 和 Chrome 里，通过 JavaScript 动态设置 iframe 的 SRC 可以避免这种阻塞情况。

②会影响页面的并行加载。

解决方案：使用js动态给iframe添加src属性值。比如：

**<iframe** id="fram"**></iframe>**

document.getelementbyid("fram").src="a2.html"

**22、webSocket如何兼容低浏览器？**

答：Adobe Flash Socket 、 ActiveX HTMLFile (IE) 、 基于 multipart 编码发送 XHR 、 基于长轮询的 XHR

**23、CSS 选择符有哪些？哪些属性可以继承？优先级算法如何计算？ CSS3新增伪类有那些？CSS新增了哪些特性？**

**CSS 选择符：**

1)      id选择器(# myid)

2)      类选择器(.myclassname)

3)      标签选择器(div, h1, p)

4)      相邻选择器(h1 + p)

5)      子选择器(ul > li)

6)      后代选择器(li a)

7)      通配符选择器( \* )

8)      属性选择器(a[rel = "external"])

9)      伪类选择器(a: hover, li:nth-child)

**可继承的样式：**

1)      font-size

2)      font-family

3)      color

4)      text-indent

**不可继承的样式：**

1)      border

2)      padding

3)      margin

4)      width

5)      height

**优先级算法：**

1)      优先级就近原则，同权重情况下样式定义最近者为准;

2)      载入样式以最后载入的定位为准;

3)      !important >  id > class > tag

4)      important 比 内联优先级高，但内联比 id 要高

**CSS3新增伪类举例：**

1)      p:first-of-type  选择属于其父元素的首个 <p> 元素的每个 <p> 元素。

2)      p:last-of-type   选择属于其父元素的最后 <p> 元素的每个 <p> 元素。

3)      p:only-of-type  选择属于其父元素唯一的 <p> 元素的每个 <p> 元素。

4)      p:only-child     选择属于其父元素的唯一子元素的每个 <p> 元素。

5)      p:nth-child(2)  选择属于其父元素的第二个子元素的每个 <p> 元素。

6)      :enabled :disabled 控制表单控件的禁用状态。

7)      :checked         单选框或复选框被选中。

**CSS3有哪些新特性？**

1)    CSS3实现圆角（border-radius），阴影（box-shadow），

2)    对文字加特效（text-shadow、），线性渐变（gradient），旋转（transform）

3)    transform:rotate(9deg) scale(0.85,0.90) translate(0px,-30px) skew(-9deg,0deg);// 旋转,缩放,定位,倾斜

4)    增加了更多的CSS选择器  多背景 rgba

5)    在CSS3中唯一引入的伪元素是 ::selection.

6)    媒体查询，多栏布局

7)    border-image

**24、如何居中div？如何居中一个浮动元素？**

答：①给div设置一个宽度，然后添加margin：0 auto属性

div{width:200px;margin:0 auto;}

②居中一个浮动元素:

设置当前div的宽度，然后设置margin-left:50%; position:relative; left:-250px;其中的left是宽度的一半。

div{

width:500px;

height:300px;

float:left;

margin-left:50%;

position:relative;

left:-250px;

background-color:#CCC;

}

**25、如何准确判断一个变量是数组类型？**

答： var array=[];

console.log(array **instanceof** Array);//true 判断引用类型的具体类型要用instanceof

console.log(typeof array);//结果为Object

JS中typeof与instanceof之间的区别：

**typeof**

typeof 是一个一元运算，放在一个运算数之前，运算数可以是任意类型。

它返回值是一个字符串，该字符串说明运算数的类型。typeof 一般只能返回如下几个结果：number,string,boolean,function,object,undefined。

我们可以使用 typeof 来获取一个变量是否存在，如 if(typeof a!="undefined"){alert("ok")}，而不要去使用 if(a) 因为如果 a 不存在（未声明）则会出错，**对于 Array,Null 等特殊对象使用 typeof 一律返回 object，**这正是 typeof 的局限性。

**instanceof**

instance：实例,例子

a instanceof b?alert("true"):alert("false"); //a是b的实例？真:假

**instanceof运算符用来判断一个构造函数的prototype属性所指向的对象是否存在另外一个要检测对象的原型链上。**

instanceof 用于判断一个变量**是否某个对象的实例**，如 var a=new Array();alert(a instanceof Array); 会返回 true，同时 alert(a instanceof Object) 也会返回 true;这是因为 Array 是 object 的子类。

**26、在使用Bootstrap框架的过程中，代码没有错误但是元素样式却无法正常显示，怎么回事？**

原因可能是导入的js文件顺序不正确。

因为导入的javaScript文件就是HTML文档的一部分，html文件在加载和执行javaScript文件时，会按照一定的顺序来进行加载和执行，如果导入的代码顺序有误，就可能会导致想要的效果无法正常显示。

27、**Ajax异步和同步的区别？**

同步的意思是当JS代码加载到当前AJAX的时候会使页面里后面的代码停止加载，页面出去假死状态，当这个AJAX执 行完毕后才会继续运行后面代码，页面假死状态解除。

而异步则这个AJAX代码运行中的时候后面的代码一样可以运行。

async这个属性默认是true（异步），当设置为false的时候，为同步。

异步时，当ajax发送请求后，在等待server端返回的这个过程中，前台会继续 执行ajax块后面的脚本，直到server端返回正确的结果才会去执行success，也就是说这时候执行的是两个线程，ajax块发出请求后（一个线程 ）和ajax块后面的脚本（另一个线程）

$.ajax({

method: "post",

url: "path",

cache:false,

async:false, //此时为同步

dataType: ($.browser.msie) ? "text" : "xml",

success: function(xmlobj){

}

});

//其实如果是同步的话，就失去了ajax的意义.。

**28、JS的单例模式。**

**单例模式：保证一个类仅有一个实例，并且提供一个访问它的全局访问点。**

（1） 闭包方式

var single = (function(){

var unique;

function getInstance(){

if(unique===undefined){

unique=new Construct();

}

return unique;

};

function Construct(){

// ... 生成单例的构造函数的代码

};

return{

getInstance:getInstance

}

})();

var t1=single;

var t2=single;

console.log(t1===t2);

（2）构造函数内部判断

function Construct(){

// 确保只有单例

if( Construct.unique !== undefined ){

return Construct.unique;

}

// 或者写成如下判断形式

// if(typeof Construct.unique==='object'){

// return Construct.unique;

// }

this.name = "ASY";

this.age="24";

Construct.unique = this;

};

var t1 = new Construct() ;

var t2 = new Construct() ;

console.log(t1===t2);

（3）通过重写构造函数

function Construct() {

// 缓存的实例

var instance = this;

// 其它内容

this.name = "ASY安森一";

this.age = "20";

// **重写构造函数**

Construct = function () {

return instance;

};

}

var t1 = new Construct() ;

var t2 = new Construct() ;

console.log(t1===t2);

**29、Vue.js的主要特点有哪些？**

答：（1） **简洁**：HTML模板+JSON数据，再创建一个Vue实例，就这么简单。

（2） **轻量**：~24kb，min+gzip，无依赖。

（3）**快速**：精确有效地异步批量DOM更新。

（4） **数据驱动**：自动追踪依赖的模板表达式和计算属性。

（5） **模块友好**：通过NPM安装，无缝融入你的工作流。

（6） **组件化**：用解耦、可复用的组件来构造界面。

**30、JQuery事件冒泡的介绍以及如何阻止事件冒泡？**

答：在一个对象上触发某类事件（比如单击onclick事件），如果此对象定义了此事件的处理程序，那么此事件就会调用这个处理程序，如果没有定义此事件处理程序或者事件返回true，那么这个事件会向这个对象的父级对象传播，从里到外，直至它被处理（父级对象所有同类事件都将被激活），或者它到达了对象层次的最顶层，即document对象（有些浏览器是window）。

如何来阻止jQuery事件冒泡？

（1）event.stopPropagation();

（2）return false;

**event.cancelBubble()属于JS的阻止事件冒泡的方法。**

**31、对BFC规范的理解？**

**32、CSS定义的权重**

答：id的权重为100，class的权重为10，标签的权重为1.

**33、什么是CSS预处理器？比较流行的CSS预处理器语言有哪些？**

答：CSS预处理器定义了一种新的语言，其基本思想是，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。

**比较流行的CSS预处理器语言有：Sass、Less、Stylus。  
34、Node.js的适用场景有哪些？**

**答：在线聊天、实时通知推送、高并发的场景。**

**35、JavaScript如何实现继承？（六种方式）**

答：（1）原型链继承

（2）构造函数继承

（3）组合继承

（4）原型式继承

（5）寄生式继承

（6）寄生组合式继承

**36、Javascript中，有一个函数，执行时对象查找时，永远不会去查找原型，这个函数是？**

答：HasOwnProperty();

**37、JavaScript里的原型、原型链？有什么特点？**

答（1）JS中每个函数都存在有一个原型对象属性prototype。并且所有函数的默认原型都是Object的实例。

（2）每个继承父函数的子函数的对象都包含一个内部属性\_proto\_。该属性包含一个指针，指向父函数的prototype。若父函数的原型对象的\_proto\_属性为再上一层函数。在此过程中就形成了原型链。

（3）原型链实现了继承。原型链存在两个问题：a 包含引用类型值的原型属性会被所有实例共享。b 在创建子类型时，无法向超类型的构造函数中传递参数。

**38、js延迟加载的方式有哪些？**

答：（1）defer属性；

（2）async属性

（3）动态创建DOM方式（用的最多）

（4）使用jQuery的getScript()方法

（5）使用setTimeout延迟方法

（6）让JS最后加载

**39、异步加载的方式有哪些？**

答：（1）<script>标签的async="async"属性

（2）<script>标签的defer="defer"属性

（3）动态创建<script>标签，加载完毕后callBack

**40、五大浏览器内核：**

答：（1）Trident

（2）Gecko

（3）Presto

（4）Webkit

（5）Blink

**41、原生JS如何实现数据的双向绑定？**

对于前端，有时候需要实现视图层和数据层的双向绑定(two-way-binding), 例如当前流行的各种框架和类库:Vue.js、Angular.js、React.js。

双向绑定就是将数据的变化绑定到UI上，同时UI的变化又和数据同步。

实现方式大概有两种：

1.手动绑定，同时使用dirty check(脏检查)去循环监听。（代表angular js）

2.前端数据劫持，使用Object.define Property()。(代表Vue js)

**Vue和React的相同点和不同点？**

1、相同点

①都支持组件化开发和VirtualDOM

②都支持props进行父子组件间数据通信

③都支持数据驱动视图，不直接操作真实DOM，更新状态数据界面就自动更新

④都支持服务器端渲染

⑤都支持native的方案，React的React Native，Vue的Weex

**42、Vue实现数据双向绑定的原理？**

**不过在不同的 MVVM 框架中，实现双向数据绑定的技术有所不同。**

**AngularJS 采用“脏值检测”的方式，数据发生变更后，对于所有的数据和视图的绑定关系进行一次检测，识别是否有数据发生了改变，有变化进行处理，可能进一步引发其他数据的改变，所以这个过程可能会循环几次，一直到不再有数据变化发生后，将变更的数据发送到视图，更新页面展现。如果是手动对 ViewModel 的数据进行变更，为确保变更同步到视图，需要手动触发一次“脏值检测”。**

**VueJS 则使用 ES5 提供的 Object.defineProperty() 方法，监控对数据的操作，从而可以自动触发数据同步。并且，由于是在不同的数据上触发同步，可以精确的将变更发送给绑定的视图，而不是对所有的数据都执行一次检测。**

访问器属性是对象中的一种特殊属性，它不能直接在对象中设置，而必须通过 defineProperty() 方法单独定义。

vue.js 是采用数据劫持结合发布者-订阅者模式的方式，通过Object.defineProperty()来劫持各个属性的setter，getter，在数据变动时发布消息给订阅者，触发相应的监听回调。

Object.defineProperty() 方法会直接在一个对象上定义一个新属性，或者修改一个对象的现有属性， 并返回这个对象。

它接收三个参数，要操作的对象，要定义或修改的对象属性名，属性描述符。重点就是最后的属性描述符。

var obj = { };

// 为obj定义一个名为 hello 的访问器属性

Object.defineProperty(obj, "hello", {

get: function () {

return sth;

},

set: function (val) {/

\* do sth \*/

}

})

obj.hello // 可以像普通属性一样读取访问器属性

访问器属性的"值"比较特殊，读取或设置访问器属性的值，实际上是调用其内部特性：get和set函数。

obj.hello // 读取属性，就是调用get函数并返回get函数的返回值

obj.hello = "abc" // 为属性赋值，就是调用set函数，赋值其实是传参



**43、display:none和visibility:hidden的区别？**

<div style="display:">显示</div>

<div style="display:none;">隐藏不占位</div>

<div style="visibility:">显示</div>

<div style="visibility:hidden;">隐藏占位</div>

<div visible="true" runat="server">显示</div>

<div visible="false" runat="server">消失不占位</div>

1）display

　　　　隐藏必须使用none值，该隐藏不占位，页面源代码可见。

　　　　显示可使用空或block或inline，意思分别为：无此属性、显示视为div（有换行符）、显示视为span（无换行符）。

　　2）visibility

　　　　隐藏必须使用hidden值，该隐藏占位，页面源代码可见。

　　　　显示可使用空或visible，意思分别为：无此属性、显示。

　　3）visible（服务器控件）

　　　　隐藏必须使用false值与runat="server"，该隐藏不占位，页面源代码不可见。

　　　　显示可使用空或true，意思分别为：无此属性、可见。

44**、GET请求和POST请求的区别？**

request获取请求参数

最为常见的客户端传递参数方式有两种：

浏览器地址栏直接输入：一定是GET请求；

超链接：一定是GET请求；

表单：可以是GET，也可以是POST，这取决与<form>的method属性值；

HTTP协议中定义了浏览器和服务器进行交互的不同方法，基本方法有4种，分别是GET，POST，PUT，DELETE。这四种方法可以理解为，对服务器资源的查，改，增，删。

1. GET：从服务器上获取数据，也就是所谓的查，仅仅是获取服务器资源，不进行修改。
2. POST：向服务器提交数据，这就涉及到了数据的更新，也就是更改服务器的数据。
3. PUT：PUT的英文含义是放置，也就是向服务器新添加数据，就是所谓的增。
4. DELETE：从字面意思也能看出，这种方式就是删除服务器数据的过程。

1、GET交互方式是从服务器上获取数据，而并非修改数据，所以GET交互方式是安全的。就像数据库查询一样，从数据库查询数据，并不会影响数据库的数据信息，对数据库来说，也就是安全的。

2、在GET交互，对同一个URL的多个请求，得到的结果是相同的。就像数据库查询，不同的数据库连接对同一个数据库表用相同条件查询时，得到的结果也是一样的。

1、POST交互是可以修改服务器数据的一种方式，涉及到信息的修改，就会有安全问题。就像数据库的更新，Update一个数据库表时，如果条件没有写对，就可能把不需要修改的数据给修改了，得到的数据就是错误的了。

2、一般的POST交互是必须要用到表单的，但是表单提交的默认方法是GET，如果改为POST方式，就需要修改表单提交时的Method。

两者的区别：

1. 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。

2. get安全性非常低，post安全性较高。但是get执行效率却比post方法好。

3. 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。

4. 对于get方式，服务器端用Request.QueryString获取变量的值，对于post方式，服务器端用Request.Form获取提交的数据。

所以综上建议：

1、包含机密信息的话，建议用Post数据提交方式；

2、在做数据查询时，可以使用Get方式；而在做数据添加、修改或删除时，建议用Post方式；

45、**一个页面从输入url到页面加载显示完成，中间都经历了什么？**

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

其中，步骤2的具体过程是：

* **浏览器缓存**：浏览器会记录DNS一段时间，因此，只是第一个地方解析DNS请求；
* **操作系统缓存：**如果在浏览器缓存中不包含这个记录，则会使系统调用操作系统，获取操作系统的记录(保存最近的DNS查询缓存)；
* **路由器缓存**：如果上述两个步骤均不能成功获取DNS记录，继续搜索路由器缓存；
* **ISP缓存：**若上述均失败，继续向ISP搜索。

**45、ES6中**[**let与const的区别**](http://www.cnblogs.com/dupd/p/6096235.html)

let与const都是只在声明所在的块级作用域内有效。

let声明的变量可以改变，值和类型都可以改变，没有限制。

const声明的变量不得改变值，这意味着，const一旦声明变量，就必须立即初始化，不能留到以后赋值。

46、原型的理解

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title></title>

</head>

<body>

<p>

当函数对象本身的属性或方法与原型的属性或方法同名的时候：<br />

1、默认调用的是函数对象本身的属性或方法；<br />

2、通过原型增加的属性或方法的确是存在的；<br />

3、函数对象本身的属性或方法的优先级要高于原型的属性或方法。

</p>

<script type="text/javascript">

function Person(){

this.name="安森一";

this.showMe=function(){

alert("名字叫安森一");

}

}

Person.prototype.name="安森二";

Person.prototype.showMe=function(){

alert("名字叫安森二");

}

Person.prototype.sayMe=function(){

alert("名字叫安森三");

}

var one=new Person();

alert(one.name);

one.showMe();

one.sayMe();

</script>

</body>

</html>

47、如何使用CSS3实现动态效果？

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title></title>

<style type="text/css">

div{

width:100px;

height:100px;

background:red;

position:relative;

animation:myfirst 3s infinite alternate;

}

@keyframes myfirst {

0% {background:red; left:0px; top:0px;}

25% {background:yellow; left:200px; top:0px;}

50% {background:blue; left:200px; top:200px;}

75% {background:green; left:0px; top:200px;}

100% {background:red; left:0px; top:0px;}

}

</style>

</head>

<body>

<div></div>

</body>

</html>

48、国内的设计大师大都喜欢用px，而国外的网站大都喜欢用em和rem，那么三者各有什么区别呢？

答：px，像素。是绝对长度单位。

em是相对长度单位，继承父级元素的字体大小。如果当前对象内文本的字体尺寸未被认为设置，则相对于浏览器的默认字体尺寸。

任意浏览器的默认字体都是16px。所有未经调整的浏览器都符合：1em=16px。

em的特点：

* 1. em的值并不是固定的。
  2. em会继承父级元素的字体大小。

rem是CSS3新增的一个相对单位（root em，根em），这个单位与em的区别在于使用rem为元素设定字体大小时，仍然是相对大小，但相对的是HTML根元素。

这个单位可谓集相对大小和绝对大小的优点于一身，通过它既可以做到只修改根元素就成比例地调整所有字体大小，又可以避免字体大小逐层复合的连锁反应。

49、详解<a>标签中href=””的几种用法。

<a>标签的最重要功能是实现超链接和锚点。

<a href="javascript:;">超链接</a>

<a href="javascript:void(0);">超链接</a> //这样写的目的是防止页面刷新和跳转，如果href为空，点击的话页面会刷新。

<a href="#">顶部</a> //#是标签内置的一个方法，代表top的作用，所以这种方法点击后返回到页面的最顶端。

<a href=”#” onclick=”js\_method();return false;”> //执行js方法，但是不发生页面跳转

例如：

<a href="#" onclick="alert('点击了超链接');">执行点击方法，并且回到顶部</a>

<a href="#" onclick="alert('点击了超链接');return false;">执行点击方法，但是不回到顶

部</a>

50、考察var和let声明的变量的作用域问题

**第一套面试题**

for(var i=0;i<5;i++){

setTimeout(function(){

console.log(i);

},0);

}

//输出 5个5

for(let i=0;i<5;i++){

setTimeout(function(){

console.log(i);

},0);

}

//输出0 1 2 3 4

**for(var i=0;i<5;i++){**

**(function(i){**

**setTimeout(function(){**

**console.log(i);**

**},0);**

**})(i);**

**}**

**//输出0 1 2 3 4 以上采用的是闭包的方法**

**第二套面试题**

var funcs = [];

for (var i = 0; i < 10; i++) {

funcs.push(function() {

console.log(i)

});

}

funcs.forEach(function(func) {

func();

});

//输出10个10

var funcs = [];

for (var i = 0; i < 10; i++) {

funcs.push(function() {

console.log(i)

});

}

funcs.forEach(function(func) {

func();

});

//输出0 1 2 3 4 5 6 7 8 9

var funcs = [];

for (var i = 0; i < 10; i++) {

funcs.push((function(value) {

return function() {

console.log(value);

};

}(i)));

}

funcs.forEach(function(func) {

func();

});

//输出0 1 2 3 4 5 6 7 8 9 利用闭包解决这个问题

51、如何用js去除字符串空格？

方法一：使用replace正则匹配的方法

去除所有空格: str = str.replace(/\s\*/g,"");

去除两头空格: str = str.replace(/^\s\*|\s\*$/g,"");

去除左空格： str = str.replace( /^\s\*/, “”);

去除右空格： str = str.replace(/(\s\*$)/g, "");

str为要去除空格的字符串，实例如下：

var str = " 23 23 ";

var str2 = str.replace(/\s\*/g,"");

console.log(str2); // 2323

方法二：使用str.trim()方法

str.trim()局限性：无法去除中间的空格，实例如下：

var str = " xiao ming ";

var str2 = str.trim();

console.log(str2); //xiao ming

同理，str.trimLeft()，str.trimRight()分别用于去除字符串左右空格。

方法三：使用jquery中的$.trim(str)方法

$.trim(str)局限性：无法去除中间的空格，实例如下：

var str = " xiao ming ";

var str2 = $.trim(str)

console.log(str2); // xiao ming

52、apply()/call()求数组最值

var numbers = [5, 458 , 120 , -215 ];

var maxInNumbers = Math.max.apply(this, numbers);

console.log(maxInNumbers); // 458

var maxInNumbers = Math.max.call(this,5, 458 , 120 , -215);

console.log(maxInNumbers); // 458

53、json字符串和json对象的转换

json字符串转json对象,调用parse方法：

var b='{"name":"2323","sex":"afasdf","age":"6262"}'//json字符串

var bToObject=JSON.parse(b);

console.log(bToObject.name);//2323

json对象转为json字符串：

var a={"name":"tom","sex":"男","age":"24"}//json对象

var aToString=JSON.stringify(a);

console.log(aToString);//{"name":"tom","sex":"男","age":"24"}

54、ES6中箭头函数与普通函数this的区别

**55、js中的数据类型有哪些？ES6中新增的数据类型有哪些？**

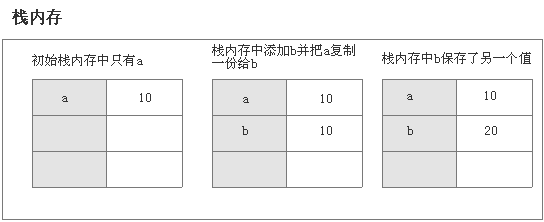
js中有六种数据类型，包括五种基本数据类型（Number,String,Boolean,Undefined,Null）,和一种引用数据类型（Object）。

（在JS中除了基本数据类型以外的都是对象，数据是对象，函数是对象，正则表达式是对象）

ES6中新增了数据类型Symbol。新的数据结构set、map。

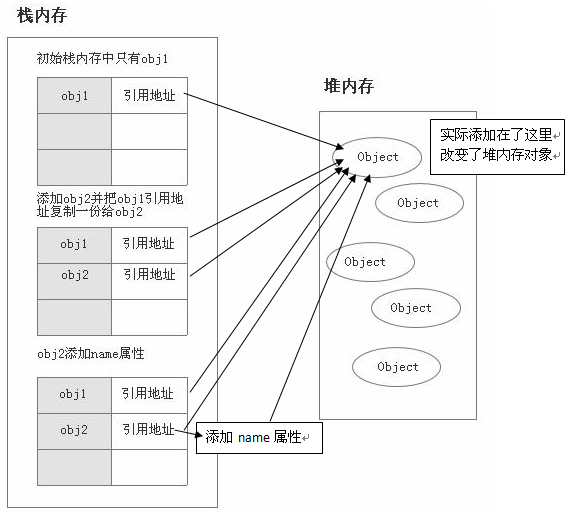
基本数据类型是指存放在栈中的简单数据段，数据大小确定，内存空间大小可以分配，它们是直接按值存放的，所以可以直接按值访问。

下图演示了这种基本数据类型赋值的过程：



引用类型是存放在堆内存中的对象，变量其实是保存的在栈内存中的一个指针（保存的是堆内存中的引用地址），这个指针指向堆内存。

引用类型数据在栈内存中保存的实际上是对象在堆内存中的引用地址。通过这个引用地址可以快速查找到保存中堆内存中的对象



基本数据类型的特点：

1.基本数据类型是按值访问的，就是说我们可以操作保存在变量中的实际的值；

2.基本数据类型的值是不可变的，任何方法都无法改变一个基本数据类型的值；

3.基本数据类型不可以添加属性和方法；

4.基本数据类型的赋值是简单的赋值；

5.基本数据类型的比较是值的比较；

6.基本类型的值在内存中占据固定大小的空间，被保存在栈内存中。

引用数据类型的特点：对象是属性和方法的集合。

1. 引用类型是按引用访问的，当查询时，我们需要先从栈中读取内存地址，然后按照指针所指向的地方，找到堆内存里面的值；

2.引用类型的值是可变的；

3.可为引用类型添加属性和方法，也可以删除其属性和方法；

4.

5.引用类型的比较是引用的比较；

6.引用类型的值是同时保存在栈内存和堆内存中的对象。

**55、HTML 表单元素有哪些？**

HTML表单用于搜集不同类型的用户输入。

文本类：

文本框：<input type = "text" />

密码框：<input type = "password" />

隐藏域：<input type = "hidden" />

type还可以为：email、url、number、range、search、date等

文本域：<textarea></textarea>

按钮类：

普通按钮：<input type = "button" value  = "text" />

提交按钮：<input type = "submit" value = "提交" />

重置按钮：<input type = "reset" value = "重置" />

图片按钮：<input type = "image" />

选择类：

单选按钮：<input type = "radio">

复选按钮：<input type = "checkbox" value = "" id = ""><label  for = ""></label>

<label></label>

下拉列表：<select>

      <option vale = "1"></option>

</select>

56、如何用jQuery禁用浏览器的前进后退按钮？

实现代码如下：

<script type="text/javascript" language="javascript">

　　$(document).ready(function() {

　　　　window.history.forward(1);

　　　　//OR window.history.forward(-1);

　　});

</script>

57、console.log(!0);输出结果为 true

console.log(!0&&10);输出结果为10

console.log(true&&10);输出结果为10

**58、用原生js写出如何获取页面所有的checkbox元素**

<script>

var inputs=document.getElementsByTagName("input");

var checkboxArray=[]; //初始化空数组，用来存放checkbox对象

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

var obj=inputs[i];

if(obj.type=='checkbox'){

checkboxArray.push(obj);

obj.checked=true;

}

}

</script>

**59、js中如何跳出forEach循环？**

forEach()方法用于调用数组的每个元素，并将元素传递给回调函数。

forEach()使用三个参数调用该函数：数组元素item、元素的索引index和数组本身array。

**注意: forEach() 对于空数组是不会执行回调函数的。**

array.forEach(function(currentValue, index, arr), thisValue)。

var arr=['a','b','c'];

arr.forEach((item,index) => {

if(item === 'b') return;

console.log(item);

});//a,c

console.log('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*');

arr.forEach((item,index) => {

if(item === 'b') throw new Error("b存在");

console.log(item);

});//a,c

**60、什么是浅拷贝和深拷贝以及如何实现深拷贝？**

什么是浅拷贝？

浅拷贝是指copy的对象和被copy的对象它们公用一个实体，也就是同时指向内存中的一个相同的地址。对于一个引用类型，如果直接将它赋值给另一个变量，由于这两个引用指向同一个地址，这时改变其中任何一个引用，另一个都会受到影响。

对于仅仅是复制了引用（地址），换句话说，复制了之后，原来的变量和新的变量指向同一个东西，彼此之间的操作会互相影响，为浅拷贝。

比如，有的地方把太阳叫“太阳”，而有的地方把太阳叫做“日头”，不管是“太阳”还是“日头”，指的都是太阳这个物体，如果某天“太阳”缺了一块儿（只说如果），那么“日头”也肯定会缺一块儿，因为它们是指的是同一物体，只是称呼不同。

什么是深拷贝？

深拷贝是指copy的对象和被copy的对象两个都是独立的、互不影响的。就像玩具厂生产出来的玩具，它们是一模一样的，都可以被买走，但是又是谁也不影响谁。

而如果是在堆中重新分配内存，拥有不同的地址，但是值是一样的，复制后的对象与原来的对象是完全隔离，互不影响，为深拷贝。

\* 一般来讲，而引用类型copy都属于浅拷贝（string类型除外），值类型的copy都属于深拷贝。

//第1种方法：递归

function deepClone(obj){

let objClone = Array.isArray(obj)?[]:{};

if(obj && typeof obj==="object"){

for(key in obj){

if(obj.hasOwnProperty(key)){

//判断ojb子元素是否为对象，如果是，递归复制

if(obj[key]&&typeof obj[key] ==="object"){

objClone[key] = deepClone(obj[key]);

}else{

//如果不是，简单复制

objClone[key] = obj[key];

}

}

}

}

return objClone;

}

let a1=[1,2,3,4],

b1=deepClone(a1);

a1[0]=2;

console.log(a1,b1);

console.log("\*\*\*\*借用JSON对象的parse和stringify实现深拷贝\*\*\*\*");

//第2种方法：还可以借用JSON对象的parse()和stringify()

function deepClone2(obj){

let \_obj = JSON.stringify(obj),

objClone = JSON.parse(\_obj);

return objClone;

}

let a2=[0,1,[2,3],4],

b2=deepClone2(a2);

a2[0]=1;

a2[2][0]=1;

console.log(a2,b2);

console.log("\*\*\*\*\*\*\*\*\*\*\*借用JQ的extend方法\*\*\*\*\*\*\*\*\*\*\*");

//第3种方法：还可以借用JQ的extend方法。

//$.extend( [deep ], target, object1 [, objectN ] )

// deep表示是否深拷贝，为true为深拷贝，为false，则为浅拷贝

// target Object类型目标对象，其他对象的成员属性将被附加到该对象上。

//object1 objectN可选。 Object类型 第一个以及第N个被合并的对象。

let a3=[0,1,[2,3],4],

b3=$.extend(true,[],a3);

a3[0]=1;

a3[2][0]=1;

console.log(a3,b3);

let obj3 = {name:'安森一',age:20,company : { name : '腾讯', address : '深圳'} };

let obj3\_extend = $.extend(true,{}, obj3); //extend方法，第一个参数为true，为深拷贝，为false，或者没有为浅拷贝。

console.log(obj3 === obj3\_extend);//false

obj3.company.name = "阿里";

obj3.name = "安森2";

console.log(obj3,obj3\_extend);

**61、数组中改变原数组的常用方法和不改变原数组的方法？**

改变原数组的：

pop()：删除最后一个元素，返回被删除的元素。如果数组为空，则不改变数组，返回undefined。

push()：向数组末尾添加元素，并返回新的长度。

shift()：将第一个元素删除并且返回删除元素。如果数组为空，则不改变数组，返回undefined。

unshift()：向数组开头添加元素，并返回新数组的长度。

reverse()：颠倒数组中元素顺序，返回该数组。

sort()：对数组进行排序。

splice():splice(start,length,item)删，增，替换数组元素，返回被删除数组，无删除则不返回

不改变原数组的：

concat()：拼接两个或多个数组，返回一个新的数组

join()：将数组中所有元素以参数作为分隔符放入一个字符串,返回值是一个字符串

slice：slice(start,end)，从start开始截取，截取到end，返回选定元素，组成一个新数组（如果没有参数, 返回整个原数组）

toString() 把数组转为字符串，返回数组的字符串形式

map,filter,forEach,some,every等不改变原数组

**62、js中哪些值转化成Boolean是false?**

""，''，0， null，undefined, NaN, false 会自动转换为false。其它的都会转换为真.。 Boolean(null)         // false  
Boolean('hello')      // true   
Boolean('0')          // true   
Boolean(' ')          // true   
Boolean([])           // true   
Boolean(function(){}) // true

**63、渐进增强和优雅降级的区别？**

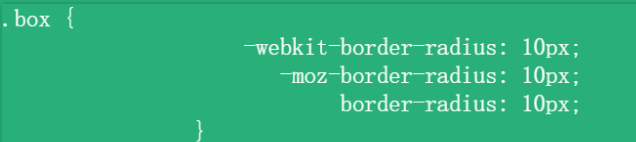
**渐进增强** (Progressive Enhancement)：针对低版本浏览器进行构建页面，保证最基本的功能，然后再针对高级浏览器进行效果、交互等改进和追加功能达到更好的用户体验

**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

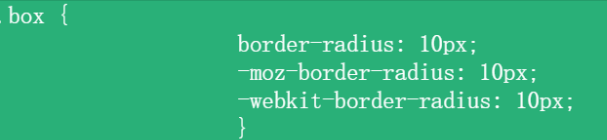
1.什么是渐进增强?

在网页开发中，渐进增强认为应该专注于内容本身。一开始针对低版本的浏览器构建页面，满足最基本的功能，再针对高级浏 览器进行效果，交互，追加各种功能以达到更好用户体验,换句话说，就是以最低要求，实现最基础功能为基本，向上兼容。以css为例，以下这种写法就是渐进增强。



2.什么是优雅降级?

在网页开发中，优雅降级指的是一开始针对一个高版本的浏览器构建页面，先完善所有的功能。然后针对各个不同的浏览器进行测试，修复，保证低级浏览器也有基本功能 就好，低级浏览器被认为“简陋却无妨 (poor, but passable)” 可以做一些小的调整来适应某个特定的浏览器。但由于它们并非我们所关注的焦点，因此除了修复较 大的错误之外，其它的差异将被直接忽略。也就是以高要求，高版本为基准，向下兼容。同样以css为例，优雅降级的写法如下。



**64、一次js请求一般情况下有哪些地方会有缓存处理？**

a. 浏览器端存储

b. 浏览器端文件缓存

c. HTTP缓存304

d. 服务器端文件类型缓存

e. 表现层&DOM缓存

**65、一个页面上有大量的图片（大型电商网站），加载很慢，你有哪些方法优化这些图片的加载，给用户更好的体验？**

a. 图片懒加载，滚动到相应位置才加载图片。

b. 图片预加载，如果为幻灯片、相册等，将当前展示图片的前一张和后一张优先下载。

c. 使用CSSsprite，SVGsprite，Iconfont、Base64等技术，如果图片为css图片的话。

d. 如果图片过大，加载时可以先加载一张缩略图，再加载大图片，替换掉缩略图。