Flex布局

Cookie、sessionstorage、localstorage的区别，如果获取localstorage的所有key

说一下BFC

了解哪些设计模式

说一下稳定和不稳定的排序有哪些

快排时间复杂度，基准元素选择的优化

如何改变this指向  
设计函数输出数组的重复元素  
undefined和null区别  
foreach和map区别  
伪数组和数组区别，转换

css实现左边固定200px右边自适应

题目：编写一个函数实现对Object，Array，Boolean，Number，String这五种主要数据类型进行值复制。  
网上搜到的参考解答：Object，Array，Boolean，Number，String分为三种情况：普通变量，Array，Object ，使用递归调用方式解题  
function clone(Obj){  
    var buf;  
    if(Obj instanceof Array){  
         buf = []; //创建一个空数组  
         var i = Obj.length;  
         while(i--){      
             buf[i] = clone(Obj[i]);  
        }  
        return buf;  
    }   
    else if (Obj instanceof Object) {  
        buf = {}; //创建一个空对象  
        for(var k in Obj){  
             buf[k] = clone(Obj[k]);  
         }  
         return buf;  
     }  
     else {  
        return Obj; //普通对象直接赋值  
     }  
}

<https://segmentfault.com/a/1190000010477169>

作者：梦游计  
链接：<https://www.nowcoder.com/discuss/34549?type=0&order=0&pos=18&page=1>  
来源：牛客网

1. 左边固定，右边自适应的两列布局，右面的 DOM 顺序在左边前面，说下实现？
2. CSS box-sizing 的值有哪些？
3. 如何阻止事件冒泡（stopPropagation）、阻止默认行为（preventDefault）？
4. 有一个字符串，里面包括 11 位手机号，如何把里面的手机号替换成别的字符？
5. Header 头 Set-cookie: http-only 是干什么用的？
6. 自我介绍
7. 讲讲你做过的项目
8. 项目中提到了 React，问：React 组件的生命周期？父子组件之间如何通信？子组件之间如何通信？
9. 前端性能优化？
10. http 缓存原理？
11. JavaScript 的垃圾回收机制？
12. CSS BFC 原理？
13. box-sizing 属性（两种盒子模型）？
14. 稳定的排序和不稳定的排序？
15. 快速排序的思路？
16. 堆排序的思路？

## 2. 请在ES5下实现一个继承，Child继承Parent。要求：Child要继承Parent的所有方法和属性。

## 4. apply和call干什么用的， 有什么区别？

## 5. css中的单位有哪些？并简述其计算方法。

**%**

百分比   
百分比不能算是一种单位，可以理解为包含数值和单位的数据，不管**离子元素最近的**父元素的该属性是什么单位，都是将 数值\*百分比+单位 计算出child的属性值。

padding和margin的值为百分比是相对于父元素的width计算的，不管是padding-left,还是padding-top

**px**

像素 (计算机屏幕上的一个点)

**ex 和 ch**

ex、ch单位与em、rem相似之处在于都依赖于font-size，但是ex、ch还依赖于font-family，基于font-specific来计算。

## 6. css的优先顺序是什么？

## 7. 如何实现js的跨域？并简述其原理

## 8. 请列举前端开发过程中，有哪些优化的点需要注意

一、减少操作量

1. 尽量减少 HTTP 请求 1) 合并文件，比如把多个 CSS 文件合成一个； 2) CSS Sprites 利用 CSS background 相关元素进行背景图绝对定位；
2. 不要在 HTML 中使用缩放图片 缩放图片并没有减少图片的容量，只是控制了图片的大小。
3. Image压缩 使用工具对图片进行压缩，保证质量的同时减少了图片的大小。
4. 减少对DOM的操作 减少对DOM的操作，减少页面的重绘。

二、提前做加载操作

1. 对域名进行预解析 例如京东的做法

<link rel="dns-prefetch" href="//misc.360buyimg.com" />

* 1

1. 预载入组件或延迟载入组件
2. 把 CSS 放到代码页上端 CSS 放到最顶部，浏览器能够有针对性的对 HTML 页面从顶到下进行解析和渲染。
3. 使用 new Image对象，对图片进行缓存

三、提升并行加载   
切分组件到多个域 ，提升服务器的响应能力

四、JavaScript和CSS优化

1. 从页面中剥离 JavaScript 与 CSS 剥离后，能够有针对性的对其进行单独的处理策略，比如压缩或者缓存策略。
2. 精简 JavaScript 与 CSS   
   使用工具压缩JavaScript和CSS文件
3. 脚本放到 HTML 代码页底部   
   减少对页面的阻塞。

五、异步加载   
使用Ajax实现异步加载，例如，滚动页面加载后面的内容，这种也比较常见。

六、避免重绘和回流

多次使用节点方法(如：appendChild)绘制页面，每次都要刷新页面一次。效率也就大打折扣了，而使用document\_createDocumentFragment()创建一个文档碎片，把所有的新结点附加在其上，然后把文档碎片的内容一次性添加到document中，这也就只需要一次页面刷新就可以了。

## 9. 请编写一个Java Script函数 parseQueryString，它的用途是把URL参数解析为一个对象，如： var url = “<http://witmax.cn/index.php?key0=0&key1=1&key2=2>”

function parseQueryString(argu){

var str = argu.split('?')[1];

var result = {};

var temp = str.split('&');

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

{

var temp2 = temp[i].split('=');

result[temp2[0]] = temp2[1];

}

return result;

}

var str = parseQueryString1('http://witmax.cn/index.php?key0=0&key1=1&key2=2');

console.log(str);

## 11. 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？

1、输入地址   
  当我们开始在浏览器中输入网址的时候，浏览器其实就已经在智能的匹配可能得url了，他会从历史记录，书签等地方，找到已经输入的字符串可能对应的 url，然后给出智能提示，让你可以补全url地址。对于google的chrome的浏览器，他甚至会直接从缓存中把网页展示出来，就是说，你还没有按下 enter，页面就出来了。

2、浏览器查找域名的 IP 地址   
1、请求一旦发起，浏览器首先要做的事情就是解析这个域名，一般来说，浏览器会首先查看本地硬盘的 hosts 文件，看看其中有没有和这个域名对应的规则，如果有的话就直接使用 hosts 文件里面的 ip 地址。   
     2、如果在本地的 hosts 文件没有能够找到对应的 ip 地址，浏览器会发出一个 DNS请求到本地DNS服务器 。本地DNS服务器一般都是你的网络接入服务器商提供，比如中国电信，中国移动。   
     3、查询你输入的网址的DNS请求到达本地DNS服务器之后，本地DNS服务器会首先查询它的缓存记录，如果缓存中有此条记录，就可以直接返回结果，此过程是递归的方式进行查询。如果没有，本地DNS服务器还要向DNS根服务器进行查询。   
    4、根DNS服务器没有记录具体的域名和IP地址的对应关系，而是告诉本地DNS服务器，你可以到域服务器上去继续查询，并给出域服务器的地址。这种过程是迭代的过程。   
    5、本地DNS服务器继续向域服务器发出请求，在这个例子中，请求的对象是.com域服务器。.com域服务器收到请求之后，也不会直接返回域名和IP地址的对应关系，而是告诉本地DNS服务器，你的域名的解析服务器的地址。   
    6、最后，本地DNS服务器向域名的解析服务器发出请求，这时就能收到一个域名和IP地址对应关系，本地DNS服务器不仅要把IP地址返回给用户电脑，还要把这个对应关系保存在缓存中，以备下次别的用户查询时，可以直接返回结果，加快网络访问。

3、浏览器向 web 服务器发送一个 HTTP 请求

拿到域名对应的IP地址之后，浏览器会以一个随机端口（1024<端口<65535）向服务器的WEB程序（常用的有httpd,nginx等）80端口发起TCP的连接请求。这个连接请求到达服务器端后（这中间通过各种路由设备，局域网内除外），进入到网卡，然后是进入到内核的TCP/IP协议栈（用于识别该连接请求，解封包，一层一层的剥开），还有可能要经过Netfilter防火墙（属于内核的模块）的过滤，最终到达WEB程序，最终建立了TCP/IP的连接。

4、服务器的永久重定向响应

　服务器给浏览器响应一个301永久重定向响应，这样浏览器就会访问<http://www.google.com/> 而非<http://google.com/>。   
　　为什么服务器一定要重定向而不是直接发送用户想看的网页内容呢？其中一个原因跟搜索引擎排名有关。如果一个页面有两个地址，就像<http://www.yy.com/>和<http://yy.com/>，搜索引擎会认为它们是两个网站，结果造成每个搜索链接都减少从而降低排名。而搜索引擎知道301永久重定向是什么意思，这样就会把访问带www的和不带www的地址归到同一个网站排名下。还有就是用不同的地址会造成缓存友好性变差，当一个页面有好几个名字时，它可能会在缓存里出现好几次。

5、浏览器跟踪重定向地址

现在浏览器知道了 “<http://www.google.com/>“才是要访问的正确地址，所以它会发送另一个http请求。这里没有啥好说的

6、服务器处理请求

经过前面的重重步骤，我们终于将我们的http请求发送到了服务器这里，其实前面的重定向已经是到达服务器了，那么，服务器是如何处理我们的请求的呢？   
　　后端从在固定的端口接收到TCP报文开始，它会对TCP连接进行处理，对HTTP协议进行解析，并按照报文格式进一步封装成HTTP Request对象，供上层使用。   
　　一些大一点的网站会将你的请求到反向代理服务器中，因为当网站访问量非常大，网站越来越慢，一台服务器已经不够用了。于是将同一个应用部署在多台服务器上，将大量用户的请求分配给多台机器处理。此时，客户端不是直接通过HTTP协议访问某网站应用服务器，而是先请求到Nginx，Nginx再请求应用服务器，然后将结果返回给客户端，这里Nginx的作用是反向代理服务器。同时也带来了一个好处，其中一台服务器万一挂了，只要还有其他服务器正常运行，就不会影响用户使用。

7、服务器返回一个 HTTP 响应

经过前面的6个步骤，服务器收到了我们的请求，也处理我们的请求，到这一步，它会把它的处理结果返回，也就是返回一个HTTP响应。HTTP响应与HTTP请求相似，HTTP响应也由3个部分构成

8、浏览器显示 HTML

WebKit渲染的过程，这个过程包括：   
解析html以 构建dom树 -> 构建render树 -> 布局render树 -> 绘制render树

9、浏览器发送请求获取嵌入在 HTML 中的资源（如图片、音频、视频、CSS、JS等等）   
　其实这个步骤可以并列在步骤8中，在浏览器显示HTML时，它会注意到需要获取其他地址内容的标签。这时，浏览器会发送一个获取请求来重新获得这些文件。   
　这些地址都要经历一个和HTML读取类似的过程。所以浏览器会在DNS中查找这些域名，发送请求，重定向等等，不像动态页面，静态文件会允许浏览器对其进行缓存。有的文件可能会不需要与服务器通讯，而从缓存中直接读取，或者可以放到CDN中。

## 12. 请简述304缓存的原理

参考:<http://blog.csdn.net/soonfly/article/details/50953814>   
如果客户端发送的是一个条件验证(Conditional Validation)请求,则web服务器可能会返回HTTP/304响应,这就表明了客户端中所请求资源的缓存仍然是**有效**的,也就是说该资源从上次缓存到现在并没有被修改过.条件请求可以在确保客户端的资源是最新的同时避免因每次都请求完整资源给服务器带来的性能问题.

**辨别条件请求**

当客户端缓存了目标资源但不确定该缓存资源是否是最新版本的时候,就会发送一个条件请求.在Fiddler中,你可以在Headers Inspector查找相关请求头,这样就可以辨别出一个请求是否是条件请求.

在进行**条件请求**时,客户端会提供给服务器一个If-Modified-Since请求头,其值为服务器上次返回的Last-Modified响应头中的日期值,还会提供一个If-None-Match请求头,值为服务器上次返回的ETag响应头的值: 

服务器会读取到这两个请求头中的值,判断出客户端缓存的资源是否是最新的,如果是的话,服务器就会返回HTTP/304 Not Modified响应,但没有**响应体.**客户端收到304响应后,就会从缓存中读取对应的资源.

另一种情况是,如果服务器认为客户端缓存的资源已经过期了,那么服务器就会返回HTTP/200 OK响应,响应体就是该资源当前最新的内容.客户端收到200响应后,就会用新的响应体覆盖掉旧的缓存资源.

只有在客户端缓存了对应资源且该资源的响应头中包含了Last-Modified或ETag的情况下,才可能发送条件请求.如果这两个头都不存在,则必须无条件(unconditionally)请求该资源,服务器也就必须返回完整的资源数据.

**为什么要使用条件请求**   
当用户访问一个网页时,条件请求可以加速网页的打开时间(因为可以省去传输整个响应体的时间),但仍然会有网络延迟,因为浏览器还是得为每个资源生成一条条件请求,并且等到服务器返回HTTP/304响应,才能读取缓存来显示网页.更理想的情况是,服务器在响应上指定**Cache-Control或Expires指令**,这样客户端就能知道该资源的可用时间为多长,也就能跳过条件请求的步骤,直接使用缓存中的资源了.可是,即使服务器提供了这些信息,在下列情况下仍然需要使用条件请求:

* 在超过服务器指定的过期时间之后
* 如果用户执行了刷新操作的话

在上节给出的图片中,请求头中包含了一个Pragma: no-cache.这是由于用户使用F5刷新了网页.如果用户按下了CTRL-F5 (有时称之为“强刷-hard refresh”),你会发现浏览器省略了If-Modified-Since和If-None-Match请求头,也就是无条件的请求页面中的每个资源.

#### 13. 请实现简单类型的两个数组去重，并把去重后的数据放入一个新的数组中。

var arr1 =[1, 2, 3, '0','2', '测试','重复',NaN, false];

var arr2 =[3,'1',NaN,'重复', false];

function removeDulMergeArr(arr1,arr2) {

var arr = arr1.concat(arr2);

arr.sort();

console.log(arr);

var temp = [];

while(arr.length>0) {

if(Object.is(arr[0],arr[1])){//ES6中严格判断两个变量是否相等，比===更严格

arr.shift();

}else{

temp.push(arr.shift());

}

}

console.log(temp);

}

removeDulMergeArr(arr1,arr2);

#### 14. 封装一个函数，作用是判断一个div在当前屏幕中，可以使用jquery操作符$（提示：一个div在屏幕中可以认为是他的左边或右边在屏幕中，以及上边或下边在屏幕中）

<!DOCTYPE html>

<html>

<head lang="en">

<meta charset="UTF-8">

<title></title>

<style>

\*{

margin: 0;

padding: 0;

}

#display{

width: 50px;

height: 50px;

position: fixed;

top:10px;

background-color: yellow;

font-size: 12px;

}

body{

height: 17000px;

width: 17000px;

}

</style>

<script>

window.onload = function(event) {

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

var ele = document.getElementById("secend");

window.onscroll = function () {

var rect = ele.getBoundingClientRect();

if (rect.top <= (window.innerHeight || document.documentElement.clientHeight) &&

rect.bottom >=0 &&

rect.right >=0 &&

rect.left<= (window.innerWidth || document.documentElement.clientWidth)) {

display.innerHTML = "div在可视范围";

} else {

display.innerHTML = "div不在可视范围";

}

}

}

</script>

</head>

<body>

<div id="display"></div>

<div id="first" style="height: 700px;width: 700px;background: green"></div>

<div id="secend" style="height: 200px;width: 200px;background: red"></div>

</body>

</html>

#### 16. 画出一个内半径为10px，外边框白色1px的纯红色圆圈，请写出css代码，className随意。

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Title</title>

<style>

.container{

width: 10px;

height: 10px;

border: 1px solid white;

border-radius:50%;//生成圆

background-color: red;

background-clip: padding-box;/\*主要考察背景颜色不填充边框，应该使用的属性\*/

}

</style>

</head>

<body>

<div class="container">

</div>

</body>

</html>

<https://blog.csdn.net/huangpin815/article/details/77802046?locationNum=8&fps=1>

<https://blog.csdn.net/wdlhao/article/details/79079660>