Flex布局

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

（二）**电话二面20180330**

Vue生命周期

Vue组件通信的方式

说一下BFC

清除浮动的方式

Js事件委托

Js中\_proto\_，说一下原型链

水平垂直居中实现

Flex布局

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

（三）**Hr面20180408**

怎么学前端

对加班的看法

绩点方面怎样

职业规划等

作者：0error,0warning,0offer  
链接：<https://www.nowcoder.com/discuss/90772?type=0&order=0&pos=6&page=0>  
来源：牛客网  
  
总得来看比较基础，本人答得一般。。13单选，2多选，2填空，8问答  
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  
跨域及解决办法  
css实现左边固定200px右边自适应  
还考了计组的知识0.1 0.2！＝0.3  
数据结构的知识哪些排序是稳定的

作者：chencong  
链接：<https://www.nowcoder.com/discuss/30678?type=0&order=0&pos=17&page=1>  
来源：牛客网  
  
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

**一面（2017.08.17，20min）**

1. 自我介绍
2. CSS 中 inline 元素可以设置 padding 和 margin 吗？
3. 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？
4. CSS box-sizing 的值有哪些？
5. JavaScript 的基本数据类型？
6. 如何判断 JavaScript 数据类型？
7. 如何给元素添加事件？
8. 如何阻止事件冒泡、阻止默认行为？
9. 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？
10. Header 头 Set-cookie: http-only 是干什么用的？
11. 还有什么想问我的？

面试官很好，说的不到位的地方也及时给指正。感觉没有什么压力。

**二面（2017.08.24，25min）**

1. 自我介绍
2. 讲讲你做过的项目
3. 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？
4. 前端性能优化？
5. http 缓存原理？
6. JavaScript 的垃圾回收机制？
7. CSS BFC 原理？
8. box-sizing 属性（两种盒子模型）？
9. 稳定的排序和不稳定的排序？
10. 快速排序的思路？
11. 堆排序的思路？
12. 还有什么想问我的？

全程我在说，面试官在听，还算不错。

**三面（2017.08.31，25min）**

1. 问了下学习成绩，考虑过读研吗？为什么选择工作？
2. 问了下项目的情况和学习经历？
3. 有没有拿到其他公司的 Offer？
4. 了解我们公司吗？用过我们的产品吗？感觉怎么样？
5. 了解我们工作时间和待遇吗？
6. 问了下家庭情况？
7. 你所期望的薪资价位？
8. 还有什么想问我的？

主要谈了一下薪资和待遇问题，说一周后会有通知。

## 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。

function Parent(age) {

this.age = age;

this.sayAge=function()

{

console.log(this.age);

}

}

Parent.prototype.sayParent = function() {

alert("this is parentmethod!!!");

}

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10

答案如下：

//寄生组合式继承

function Child(){

Parent.call(this);

}

function inheritPrototype(subType,superType) {

var prototype = object(superType.prototype); //创建对象

prototype.constructor = subType; //增强对象

subType.prototype = prototype; //指定对象

}

function object(o) {

function F() {}

F.prototype = o;

return new F();

}

inheritPrototype(Child,Parent);

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18

## 3. JS怎么判断变量的类型？

(1) 使用typeof检测   
当需要变量是否是number, string, boolean, function, undefined, json类型时，可以使用typeof进行判断。   
arr, json, nul, date, reg, error 全部被检测为object类型，，其他的变量能够被正确检测出来。

问题：1.其他变量是判断不出类型的，包括null。   
2.还有，typeof是区分不出array和json类型的。因为使用typeof这个变量时，array和json类型输出的都是object。

(2) 使用instanceof检测   
instanceof 运算符与 typeof 运算符相似，用于识别正在处理的对象的类型。与 typeof 方法不同的是，instanceof 方法要求开发者明确地确认对象为某特定类型。例如：

function Person(){

}   
var Tom = new Person();   
console.log(Tom instanceof Person); // true

问题：1.num, str和bool没有检测出他的类型，但是我们使用下面的方式创建num，是可以检测出类型的：

var num = new Number(123);   
var str = new String(‘abcdef’);   
var boolean = new Boolean(true);   
2.同时，我们也要看到，und和nul是检测的Object类型，才输出的true，因为js中没有Undefined和Null的这种全局类型，他们und和nul都属于Object类型，因此输出了true。//输出是false

(3). 使用constructor检测   
constructor本来是原型对象上的属性，指向构造函数。但是根据实例对象寻找属性的顺序，若实例对象上没有实例属性或方法时，就去原型链上寻找，因此，实例对象也是能使用constructor属性的。   
问题：(1)除了undefined和null，其他类型的变量均能使用constructor判断出类型。   
(2)不过使用constructor也不是保险的，因为constructor属性是可以被修改的，会导致检测出的结果不正确，例如：

function Person(){

}   
function Student(){

}   
Student.prototype = new Person();   
var John = new Student();   
console.log(John.constructor==Student); // false   
console.log(John.constructor==Person); // true   
在上面的例子中，Student原型中的constructor被修改为指向到Person，导致检测不出实例对象John真实的构造函数。

(3)同时，使用instaceof和construcor,被判断的array必须是在当前页面声明的！

(4)使用Object.prototype.toString.call   
注意：错误写法：Object.prototype.toString().call(error)；，toString后面没括号

Object.prototype.toString.call(变量)输出的是一个字符串(例如: ‘[object Number]’ )，字符串里有一个数组，第一个参数是Object，第二个参数就是这个变量的类型，而且，所有变量的类型都检测出来了，我们只需要取出第二个参数即可。或者可以使用Object.prototype.toString.call(arr)==”object Array”来检测变量arr是不是数组。

缺点：IE6下undefined,null均为**Object**

## 4. apply和call干什么用的， 有什么区别？

用途：在**特定的作用域**中调用函数，实际上等于设置函数体内this对象的值。   
apply和call的作用相同，区别在于接收参数的方式不同。

foo.call(this,arg1,arg2,arg3) == foo.apply(this, arguments)==this.foo(arg1, arg2, arg3)

* 1

1.call的使用

语法

call([thisObj[,arg1[, arg2[, [,.argN]]]]])

参数

thisObj 可选项。将被用作当前对象的对象。

arg1,arg2, , argN 可选项。将被传递方法**参数序列**。

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。   
2.apply()

apply与call的功能几乎一样，第一个参数意义都一样，只是第二个参数有点不同apply传入的是一个**参数数组**，也就是将多个参数组合成为一个数组传入，call从第二个参数开始，依次传值给调用函数的参数

## 5. css中的单位有哪些？并简述其计算方法。

作者：Ymuyi   
链接：<http://www.jianshu.com/p/65ef4ea631e9>   
來源：简书

**%**

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)

**em**   
em 单位转为像素值，取决于他们使用的字体大小。 此字体大小受从父元素继承过来的字体大小，除非显式重写与一个具体单位。

相对于当前对象内文本的字体尺寸。例如：   
1em 等于当前的字体尺寸；   
2em 等于当前字体尺寸的两倍。   
如果某元素以 12pt 显示，那么 2em 是24pt。   
在 CSS 中，em 是非常有用的单位，因为它可以自动适应用户所使用的字体。

**rem**   
使用rem同em一样皆为相对字体大小单位，不同的是rem相对的是HTML根元素。   
rem中的“r”代表“root”，这意味着设置当前元素的字体大小的基准为**根元素**，大多数情况下，我们会设置在html元素上。

<style>

html {font-size: 12px;}

div {font-size: 1.5rem;}

</style>

<body>

<div>Test-01 (12px \* 1.5 = 18px)

<div>Test-02 (12px \* 1.5 = 18px)

<div> Test-03 (12px \* 1.5 = 18px) </div>

</div>

</div>

</body>

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11

当然，rem单位不仅应用在字体上，还可以实现到CSS 网格系统中。   
**不常见**

**vh 和 vw**

在进行响应式布局时，我们常常会使用百分比来布局，然而CSS的百分比不总是解决每个问题的最佳方案，CSS的宽度相对于离它最近的父元素的宽度。 如果你想使用**视口的宽度、高度**而不是父元素的宽高，可以使用vh和vw单位。

1vh = viewportHeight \* 1/100;

1vw = viewportWidth \* 1/100;

* 1
* 2

使用vh、vw就可以保证元素的宽高适应不同设备。

**vmin 和 vmax**

vw和vh对应于viewport的width和height，而vmin和vmax分别对应于width、height中的最小值和最大值，例如如果浏览器的宽/高被设置为1000px/600px，那么

1vmin = 600 \* 1/100;

1vmax = 1000 \* 1/100;

* 1
* 2

**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。 引用w3C规范：

ex unit Equal to the used x-height of the first available font. [CSS3-FONTS]The x-height is so called because it is often equal to the height of the lowercase “x”. However, an ‘ex ’ is defined even for fonts that do not contain an “x”. The x-height of a font can be found in different ways. Some fonts contain reliable metrics for the x-height. If reliable font metrics are not available, UAs may determine the x-height from the height of a lowercase glyph. One possible heuristic is to look at how far the glyph for the lowercase “o” extends below the baseline, and subtract that value from the top of its bounding box. In the cases where it is impossible or impractical to determine the x-height, a value of 0.5em must be assumed.   
ch unit Equal to the used advance measure of the “0” (ZERO, U+0030) glyph found in the font used to render it.

用一副图来解释这两种单位的含义：

这两种单位，有许多用途，大部分是用于印刷时的微调整。例如，sup、sub元素分别显示上标和下标，但是我们可以使用position和bottom模拟：

<style type="text/css">

body { margin: 0; padding:0;}

.sup { position: relative; bottom: 1ex;}

.sub { position: relative; bottom: -1ex;}

</style>

<div>

AaB<span class="sup">b</span>

CcXxD<span class="sub">d</span>EeFf

</div>

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9

其他

in

英寸

cm

厘米

mm

毫米

pt

磅 (1 pt 等于 1/72 英寸)

pc

12 点活字 (1 pc 等于 12 点)

## 6. css的优先顺序是什么？

!important，使用此标记的CSS属性总是最优先的。   
!important->id->class->tag   
important 比 内联优先级高,但内联比 id 要高；如果两个规则的特殊性相同，那么后定义的会覆盖先定义的；

情况一：1.多重样式优先级   
多重样式（Multiple Styles）：如果外部样式、内部样式和内联样式同时应用于同一个元素，就是使多重样式的情况。

一般情况下，优先级如下：

（外部样式）External style sheet <（内部样式）Internal style sheet <（内联样式）Inline style

注意：有个例外的情况，就是如果外部样式放在内部样式的后面，则外部样式将覆盖内部样式。

<style type="text/css">

/\* 内部样式 \*/

h3{color:green;}

</style>

<!-- 外部样式 style.css -->

<link rel="stylesheet" type="text/css" href="style.css"/>

* 1
* 2
* 3
* 4
* 5
* 6
* 7

内联样式表的权值最高 1000；   
ID 选择器的权值为 100   
Class 类选择器的权值为 10   
HTML 标签选择器的权值为 1

(1).css优先级

是由四个级别和各个级别出现的次数决定的，值从左到右，左面的最大，一级大于一级

(2).优先级算法

每个规则对应一个初始四位数：0,0,0,0

若是行内样式优先级，则是1,0,0,0，高于外部定义

<div style=”color: red”>sjweb</div>

* 1

若是ID选择符，则分别加0,1,0,0

若是类选择符，**伪类**选择符，属性选择符，则分别加0,0,1,0

若是元素选择器，**伪元素**选择器，则分别加0,0,0,1

3.需要注意

!important的优先级是最高的，但出现冲突时则需比较”四位数“

优先级相同时，则采用就近原则

继承得来的属性，其优先级最低

## 7. 如何实现js的跨域？并简述其原理

(1)jsonp

JSONP 原理是加载一个 script，并执行一段回调 JS ，因为加载 JS 不需要遵循同源策略。使用这种技术服务器会接受回调函数名作为请求参数，并将JSON数据填充进回调函数中去。

但由此也带来了JSONP的一些问题：

* 无法发送特定的头部
* 只能是 GET 请求
* 无法发送 body

(2)window.postMessage

postMessage是HTML5新增在window对象上的方法，目的是为了解决在父子页面上通信的问题。

postMessage方法接受两个参数：   
message: 要传递的对象，只支持**字符串**信息，因此如果需要发送对象，可以使用JSON.stringify和JSON.parse做处理   
targetOrigin: 目标域，需要注意的是协议，端口和主机名必须与要发送的消息的窗口一致。如果不想限定域，可以使用通配符“\*”,但是从安全上考虑，不推荐这样做。

页面源中监听message事件就能正常获取消息了。其中，MessageEvent对象有三个重要属性：data用于获取数据，source用于获取发送消息的窗口对象，origin用于获取发送消息的源。

(3)document.domain

目前，很多大型网站都会使用多个子域名，而浏览器的同源策略对于它们来说就有点过于严格了。如，来自www.a.com想要获取document.a.com中的数据。只要基础域名相同，便可以通过修改document.domain为基础域名的方式来进行通信，但是需要注意的是协议和端口也必须相同。

(4)window.name

window.name这个全局属性主要是用来获取和设置窗口名称的，但是通过结合iframe也可以跨域获取数据。我们知道，每个iframe都有包裹它的window对象，而这个window是最外层窗口的子对象。所以window.name属性就可以被共享。

window对象有一个name属性，该属性有一个特征：即在一个窗口的生命周期内，窗口载入的所有的页面都是共享一个window.name的，每一个页面对window.name都有读写的权限，window.name是持久的存在于一个窗口载入的所有页面中的，并不会因为新的页面的载入而被重置。

源页面新建iframe框架，重定向到目标页面，然后使用iframe.contentWindow.name获取数据。

数据页面中要存放的数据需要存储在window.name 属性中

\*\*var data={msg:"hello world"};

window.name=JSON.stringify(data);//name属性只支持字符串，支持最大2MB的数据

* 1
* 2

(5)CORS

CORS的核心思想是通过一系列新增的HTTP头信息来实现服务器和客户端之间的通信。所以，要支持CORS，服务端都需要做好相应的配置，这样，在保证安全性的同时也更方便了前端的开发。

在服务器端返回允许跨域访问的头。

## 8. 请列举前端开发过程中，有哪些优化的点需要注意

页面优化的方法非常多，最好能够对这些优化方案进行分类，这些方案最好能够结合实际开发遇到的问题来表述。

优化的方案

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);

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15

## 10. 请简述web storage和cookie的区别。

1 、**与服务器交互**   
Web Storage 中的数据仅在存在本地，不与服务器发生交互。   
Cookie 中的数据会在浏览器和服务器中来回传递。   
2 、**存储空间**   
Web Storage 存储空间更大，可以达到 5M ；Cookie 数据大小不超过 4KB 。   
3 、 **接口**   
Web Storage 提供 setItem ， getItem ， removeItem ， clear 等方法   
Cookie 需要自己封装 setCookie ， getCookie方法   
4 、**跨域问题**   
cookie 需要指定作用域，不可以跨域调用，但 Web Storage 可以跨域调用   
5、 **存储时间**   
cookie 中的数据在过期时间之前均有效， Web Storage 则不同， sessionStorage 中的 数据在当前浏览器窗口关闭后自动删除， localStorage 持久存储数据，除非主动删除数据

注： 但 Cookie 是不可以或缺的： Cookie 的作用是与服务器进行交互，作为 HTTP 规范的一部分而存在 ，而 Web Storage 仅仅是为了在本地 “ 存储 ” 更大容量数据而生。

## 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);

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17

#### 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>

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23
* 24
* 25
* 26
* 27
* 28
* 29
* 30
* 31
* 32
* 33
* 34
* 35
* 36
* 37
* 38
* 39
* 40
* 41
* 42
* 43
* 44
* 45
* 46
* 47
* 48
* 49
* 50
* 51
* 52
* 53

#### 15. 请简述css单位rem，以及rem适合使用的场景。

rem是指相对于HTML根元素的字体大小的单位。   
rem适用于响应式布局，每个元素的宽高、文字大小、行距、补白等一切可以使用长度单位的地方均可使用。

#### 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>

#### 17. 试举例：3个行内元素与块级元素

行内元素：a ,label ,input, textarea, select ,img,span，i ,code ,q ,sub, sup   
块级元素：div ul ol li h1~h6 p form table

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

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