# 面试中总会遇见各种奇葩问题，永远打在你知识体系覆盖不到的地方。

So 多多积累，不要放弃，机会是随机碰到的。

# 总体要求：（记住带几张白纸和笔）

需要注意：对于各种面试官，他们关注的点也不一样，一般来说年龄较大的面试官问的都是一些比较老的知识，相对比较基础，要求对知识的掌握程度比较底层，他们一般不再开发代码，所以更倾向于问一下原理性的东西，具体怎么样他们也不知道，很大程度上是凭感觉来面试的。而对于较为年轻的面试官，更关注兴趣，代码的编写，或者是博客中提到的问题，他们更可能会针对你的笔试题目一道一道去分析，去深入，这就要求知识全面，各方面都有接触过，需要有实际的项目来支撑。

面试的时候要注意，气势不能弱，不要用“不能，不会”等字眼，可以大胆说出自己的想法/观点，如果没有思路，可以说“我之前没有这方面的深入研究，我们可以探讨一下”，需要注意的是，笔试要答出水平，给人一个优秀的第一印象，字迹工整，逻辑清晰，如果有代码题，可以只写关键步骤或者是伪代码，不要涂涂抹抹，卷面很乱，这样会影响下面的面试，可能就直接忽略你的答案了

前端的要求包括：

1. HTML5+CSS3

2. http协议

3. js基本知识

4. vue套装，项目中使用过的实例

5. 项目中使用过的ES5和ES6的新语法

6. 熟悉webpack等工具的使用

7. 能够了解一些简单排序，去重算法的原理，并可支持一些变换形式的问题

8. 前端安全，对于一些具有保密要求的公司来说会有这个需求

注意：语言本身没有复杂的地方，但是思维过程一定要清晰

扩展项目：

7. node的实际项目经验

8. 有原生app开发功底

9. 熟悉某种后端语言

# 参考文档：

<http://blog.jobbole.com/78346/> 40个重要的HTML5面试题及答案

<https://mp.weixin.qq.com/s/9yA8RZxAqag0du9RzRj8sw> 面试的信心来源于过硬的基础

<https://www.zhihu.com/question/20391668> 前端工程师应该对 HTTP 了解到什么程度？从哪些途径去熟悉更好？

《http权威指南》完整版

<http://mp.weixin.qq.com/s/9Uq8JOnQjVlEnXM-OrF-PA> 如果面试官问你：从输入URL到页面加载发生了什么

<http://www.ruanyifeng.com/blog/2014/09/illustration-ssl.html> 图解SSL/TLS协议-阮一峰

<https://segmentfault.com/a/1190000011145364> 前端常见跨域解决方案（全）

<https://mp.weixin.qq.com/s/3DPL_TWBkPTj9is5YR_X4g> 面试分享：2018阿里巴巴前端面试总结(题目+答案)

<https://mp.weixin.qq.com/s/6Mo5csEDVKMq4-v6Yi8ZPQ> 前端知识点总结——Vue

<http://mp.weixin.qq.com/s/GFUivceIlbjVDfcJibnVOg> 从浏览器多进程到JS单线程，JS运行机制最全面的一次梳理

<https://mp.weixin.qq.com/s/T0EIVvXqnncFG_KA1SRKZg> 阿里、网易、滴滴共十次前端面试碰到的问题

<https://mp.weixin.qq.com/s/6LnhkX63XaC05aOfHFO2zQ> 面试分享：一年经验初探阿里巴巴前端社招

<http://mp.weixin.qq.com/s/Qz0UjI3KSy4o0RRa9qqlfg> 一篇文章搞定面试中的变量提升

<http://mp.weixin.qq.com/s/d3tHHh1JQlqiXv2aP0lmWw> 关于移动端开发遇到的坑

<http://blog.csdn.net/qq_31628337/article/details/71056294> JS：事件循环机制、调用栈以及任务队列

<https://www.cnblogs.com/hfxm/p/5547922.html> JS作用域面试题总结

<http://mp.weixin.qq.com/s/IrzqWuHMninsRizJk_dfAg> 扯不完的 JS 分号问题

<https://www.cnblogs.com/cuew1987/p/4075027.html> 使用Object.create()实现继承

<http://blog.csdn.net/js_admin/article/details/71012367> javascript中实现继承的三种方式（一看就懂）

<http://blog.csdn.net/qq_37467034/article/details/78311591> this指向详解及改变它的指向的方法

<http://blog.csdn.net/qq_37467034/article/details/78311591> this指向详解及改变它的指向的方法

# 基础知识

## HTML

### ****1. SGML和HTML，XML和HTML他们之间有什么关系****

**SGML（标准通用标记语言）是一个标准，规定怎么去指定文档标记，HTML是被SGML描述的标记语言，HTML是被用SGML描述的标记语言**

**所以利用SGML创建了HTML参照和必须遵守的DTD（文档类型定义），在html文档首部的“DOCTYPE”属性用来解析目标的DTD**

**SGML是所有类型的父类，较旧的HTML利用SGML，HTML4.0使用派生自XML的XHTML**

### **2. 什么是HTML5**

**HTML5是最新的HTML标准，他目标是提供所有的内容而不需要任何的插件，这些内容包括动画，视频，**富GUI等，HTML5是w3c和WHATWG合作输出的

### 3. 为什么HTML5里面不需要DTD(document type definition)

因为HTML5没有使用SGML或者XHTML,这是个全新的东西，所以不需要写DTD，但是需要放置<!doctype html>来告诉浏览器这是h5文档

### 4. 如果不放入<!doctype html>，h5还会工作吗

不会，因为浏览器将不能识别它是HTML文档，同时H5标签不能正常工作

### 5. 哪些浏览器支持H5

几乎所有浏览器目前都已经支持了，safari，chrome，firefox，opera，ie

### 6. H5的页面结构同HTML4或者更前的HTML有什么区别

一个典型的web页面包含头部，脚部，导航，中心区域，侧边栏，在HTML4中要实现这个，我们可能需要使用DIV标签。

但是在h5中专门为这些区域创建了元素名称，可读性变得更好。

* <header>:页面头部
* <footer>:脚部
* <nav>: 页面导航元素
* <article>: 自包含的内容
* <section>:使用内部article来定义区域或者是把分组内容放在区域里面
* <aside>: 代表页面的侧边栏内容

### 7. HTML5中的datalist是什么？

HTML5中的datalist元素有助于提供文本框自动完成特性



### ****8. HTML5中什么是不同的新的表单元素类型？****

**新的表单元素大约有10个，**

**Color，date，datetime-local，email，time，url，range，telephone，number，search**

### ****9. H5中什么是输出元素？output****

**当我们需要计算两个输入的和值到一个标签中的时候我们需要输出元素**



### ****10. 什么是SVG（Scalable Vector Graphics可缩放矢量图形）****

**Svg表示可缩放矢量图形，这是基于文本的图形文本，使用文本，线条，点来进行图像绘制，更加轻便，显示更加迅速**



### ****11. H5中的canvas是什么****

**Canvas是HTML中可以绘制图形的区域**

**使用canvas的步骤是：**

* **定义canvas区域**
* **获取访问canvas上下文区域**
* **绘制图形**



### ****12. canvas和svg的区别是什么****

**二者都可以在浏览器中绘制图形，但是还是有区别的**

* **任何使用Svg绘制的形状都能被记忆和操作，浏览器可以再次显示，而canvas绘制完成后不能访问像素和操作**
* **Svg对于创建图形比如cad软件比较良好，用户可以操作。Canvas在适合于动画和游戏**
* **Svg需要记录坐标，所以比较缓慢，cancas不需要记录，渲染比较快**
* **Svg可以使用绘制对象的相关事件处理，但是canvas不能使用绘制对象的方式去处理，因为没有参考（绘制对象指的是把元素部分提前组合一下）**
* **Svg和分辨率是没有关系的（矢量图），canvas和分辨率是有关系的**

### **13. 如何使用canvas和svg画一个矩形**

**Svg :**

**<svg xmlns="http://www.w3.org/2000/svg" version="1.1">**

**<rect style="fill: rgb(0, 0, 255); stroke-width: 1px; stroke: rgb(0, 0, 0);" height="[object SVGAnimatedLength]" width="[object SVGAnimatedLength]">**

**</rect>**

**Canvas:**

**var c=document.getElementById("mycanvas");**

**var ctx=c.getContext("2d");**

**ctx.rect(20,20,150,100);**

**ctx.stroke();**

### 14. 什么是web workers？为什么我们需要他们

考虑这种情况，当页面中的一个动作时间特别长比如一个10000000次的循环，会造成页面阻塞甚至报错

Web worker的作用就是帮我们异步执行js文件，不用等到这个循环结束就可以执行下面的动作

### 15. Web worker线程的限制是什么？

Web worker线程不能修改HTML元素，全局变量和window.Location一类的窗口属性。可以自由使用js数据类型，XMLHTTPREQUEST调用等

### 16. 我们如何在javascript中创建一个worker线程

创建一个worker线程，我们需要通过js文件名创建worker对象

Var worker = new worker(“MyHeavyProcess.js”)

我们需要使用“PostMessage”发送信息给worker对象

Worker.postMessage()

当worker线程发送数据的时候，我们在调用结束的时候通过onmessage时间获取：

Worker.onmessage = function(e){

Document.getElementById(“txt1”).value = e.data;

}



### 17. 如何中止web worker

W.terminate();

### 18. 为什么我们需要HTML5的服务发送事件

互联网的普遍需求是从服务器更新，有时候我们经常需要从服务器获取最新的数据



现在通常使用的是pull的方法去服务器抓取区间数据，这种方案挺好，但是这样一来，调用了很多的网络请求，同时也增加了服务器的负担

所以相比与pull，我们希望有中push的方案，也就是当服务器更新的时候，发送更新到浏览器客户端，

所以首先浏览器需要连接会发送更新的服务器资源，比如我们有一个index.Aspx会发送股票更新，因此连接该页面，我们需要附加时间来源对象，如下

Var source = new EventSource(“index.aspx”);

当我们将要接收服务器发送的更新信息时，我们需要附加功能，我们需要附加功能到“onmessage”事件

source.onmessage = function (event) {

document.getElementById("result").innerHTML += event.data + "<br>";

};

现在来自服务端，我们需要去发送事件，下面是一些用命令需要从服务端发送的重要事件列表

|  |  |
| --- | --- |
| Event | Command |
| 发送数据到客户端 | data : hello |
| 告诉客户端10s内重试 | retry : 10000 |
| 提出具体事件与数据 | event : successdata : You are logged in. |

因此，举例说明，如果你想下面的ASP.NET代码一样发送数据，请标记内容类型设置给文本/事件

Response.ContentType="text/event-stream";

Response.Expires=-1;

Response.Write("data: " + DateTime.Now.ToString());

Response.Flush();



### 19. HTML5中的本地存储概念是什么？

很多时候我们会保存用户的信息到电脑上面，比如有的时候断网了，用户希望能把表单信息存储到本地，这样等网络恢复的时候就可以直接发送到服务器

浏览器中拥有的存储被叫做“local storage”

### 20. 我们怎么从本地存储中添加和移除数据？

数据添加到本地存储时候采用的是键值对，

添加数据：

LocalStorage.setItem(“Key001”,”India”)

获取数据：

var country = localStorage.getItem(“Key001”);

如果想存储json形式，用json.stringify方法

LocalStorage.setItem(“i001”,JSON.stringify(country))

### 21. 本地存储的生命周期是什么？

本地存储没有生命周期，它将保留直到用户从浏览器清除或者使用js移除

### 22. 本地存储和cookie（存储在用户本地终端上面的数据）之间的区别是什么

* 从客户端和服务端的角度，cookie在客户端和服务端都能访问数据，cookie通过每个请求发送到服务端，local storage只有本地浏览器可访问数据，服务器不能直接访问本地存储
* 从大小的角度，每个cookie最大4095个字节，而localstorage每个域5m
* 从有效期的角度，cookies有有效期，所以在有效期后cookie和cookie数据会被删除，localstorage没有过期数据，需要自己手动清除

### 23. 什么是事务存储？我们如何创建一个事务存储？sessionstorage

会话存储和本地存储类似，但数据在会话中有效，简而言之，数据在关闭浏览器的时候就被删除了。

如下示例，刷新浏览器的时候变量值会增加，关闭浏览器的时候变量会重置为0

if(sessionStorage.clickcount){sessionStorage.clickcount=Number(sessionStorage.clickcount)+1;}

else{sessionStorage.clickcount = 0;}

### 24. 本地存储和事务存储之间的区别是什么？

本地存储数据持续永久，但是会话在浏览器打开时候有效，浏览器关闭时候重置

### 25. 什么是webSQL？

webSQL是一个在浏览器客户端的结构关系数据库，这是浏览器内的本地RDBMS（关系型数据库），可以使用SQL查询

### 26. websql是HTML5的一个规范吗？

不是，虽然很多人把它标记为h5，但是他不是h5规范的一部分，这个规范是基于SQLlite的

### 27. 我们如何使用websql



### 28. HTML中的应用缓存是什么？

应用缓存实现的最终是用户的离线浏览，如果网络连接不可用，页面应该来自浏览器缓存，应用缓存可以帮我们指定哪些文件需要缓存，哪些不需要

### 29. 我们如何实现应用缓存



### 30. 我们如何刷新浏览器的应用缓存？

应用缓存通过变更“#”标签后 的版本号而被移除



### 31. 应用缓存中的回退是什么？

应用中的回退帮助你在制定在服务器不可访问的时候，会显示某文件。如在下面的manifest文件中，我们说如果谁敲击了”/home”，同时服务器不可到达，”homeoffline.html”文件应送达



应用缓存中的网络是什么？

网络命令描述不需要缓存的文件，如下代码，我们说home.aspx永远不应该被缓存或离线访问



## ****关于CSS****

### 1. css（cascading style sheets级联样式表）中的选择器是什么，有哪些？哪些属性可以继承？css优先级算法怎么计算

选择器在应用一个样式的时候，用于选择元素，css选择器包括：

类别选择器：.demoDiv

标签选择器：p

ID选择器：#demoDiv

后代选择器： .father.child{

子选择器：#links > a,子选择器与后代选择器的区别在于子选择器只选择直接后代，后代选择器选择的是所有的后代

伪类选择器：a:link，a:visited，a:hover（用于鼠标悬停效果等）

通用选择器：\* 选择这一级下面所有的元素

群组选择器：p, td, li 使用逗号分隔

属性选择器：判断html标签的某个属性是否存在的方法来选择

伪类元素选择器：　:before 所有伪元素选择器都必须放在出现该伪元素的选择器的最后面

Ui元素状态伪类选择器： :enabled， :read-only

可继承的属性：font-size,font-family,color

不可继承的样式: border, padding, margin, width, height

优先级（就近原则）：!important>[id>class>tag](!important比内联样式 的优先级高)

关于css 的优先级算法，

标签选择器：1，

Class选择器：10

Id选择器：100，

元素内联样式选择器：1000

1. !important声明的样式优先级最高，如果冲突再进行计算

2. 如果优先级相同，则选择最后出现的样式

3. 继承得到的样式的优先级最低

### 2. 四种定位的区别

* Static是默认值
* Relative 相对定位，相对于自身原有的位置进行偏移，仍处于标准文档流之中
* Absolute 绝对定位 相对于最近的已定位的祖先元素，已定位指的是position不为static的元素，如果无法定位祖先元素，则以body元素为参照标准偏移，完全脱离文档流
* Fixed 固定定位的元素会相对于视窗来定位，始终保持自己在视窗中的位置，一个固定元素不会保留它原本在页面应有的空隙

### 2. CSS中使用列布局是什么？

Css列布局可以使分隔文本为列，





### 3. 简要解释一下盒子模型

盒子模型是在html元素周围定义的border，padding，margin的矩形空间

### 4. 你能解释一些css3中的文本效果吗？

回答两个css的文本效果：

阴影文本效果：text-shadow: 5px 5px 5px #FF0000;

文字包装效果： word-wrap:break-word; 文本换行，比如英文显示中，单词不能从中间断开的情况

### **5. 介绍一下标准的css盒子模型，与低版本IE的盒子模型有什么不同？**

标准盒子模型： 宽度=内容的宽度（content）+border+padding+margin

低版本IE盒子模型（怪异模式）：宽度=内容宽度（content+border+padding）+margin

### 6**. box-sizing属性**

用来控制元素的盒子模型的解析模式，默认为content-box

Content-box:w3c的标准盒子模型，设置元素的height/width属性指的是content部分的高/宽

Border-box:IE传统盒子模型，设置元素的height/width属性指的是border+padding+content部分的高/宽

### 7. css3新增伪类有哪些？

P:first-of-type 选择属于其父元素的首个元素

P:last-of-type 选择属于其父元素的最后元素

P:only-of-type 选择属于其父元素的唯一元素

P:only-child 选择属于其父元素的唯一子元素

P:nth-child(2) 选择属于其父元素的第二个子元素

:enabled:disabled 表单控件的禁用状态

:checked 单选框或复选框被选中

### 8. 如何居中div？如何居中一个浮动元素？如何让绝对定位的div居中

div{

border:1px solid red;

margin:0 auto;

height:50px;

with:80px;

}

float{

border:1px solid red;

float:left;

position:absolute;

width: 200px;

height: 100px;

left: 50%;

top: 50%;

margin: -50px 0 0 -100px;

}

absolute{

border: 1px solid black;

position: absolute;

width:200px;

height:100px;

margin: 0 auto;

left:0;

right: 0;

}

### 9. display有哪些值？说明他们的作用？

Inline（默认）-内联

None 隐藏

Block 块显示

Table 表格显示

List-item 项目列表

Inline-block 行内块

### 10. position的值

Static（默认） 按照正常文档流进行排列

Relative（相对定位） 不脱离，参考自身静态位置，通过top,bottom,left,right定位

Absolute（绝对定位） 参考据其最近一个不为static的父级元素通过top，bottom，left，right定位

Fixed（固定定位） 所固定的参照对象是可视窗口

### 11. css3有哪些新特性

rgba和透明度

background-image background-origin(content-box/padding-box/border-box) background-size background-repeat

word-wrap (对长的不可分割单词换行) word-wrap：break-word

文字阴影：text-shadow： 5px 5px 5px #ff0000;（水平阴影 垂直阴影 模糊距离 阴影颜色）

Font-face属性：定义自己的字体

圆角（边框半径）： border-radius创建圆角

边框图片：border-image:url(border.png) 30 30 round

盒阴影：box-shadow:10px 10px 5px #888888

媒体查询：定义两套css，当浏览器的尺寸变化时采用不同的属性

### 12. 请解释一下css3的flexbox（弹性盒布局模型），以及适用场景

该布局模型的目的是提供一种更加高效的方式来对容器中的条目进行布局，对齐和分配空间。在传统的布局方式中，block布局是把块在垂直方向上从上到下依次排列的；而inline布局则是在水平方向上来排列。弹性盒布局并没有这样内在的方向限制，可以有开发人员自由操作

适用场景： 弹性布局适合于移动前端开发，在android和ios上也完美支持

### 13. 用纯css创建一个三角形的原理是什么

首先把元素的宽度高度设置为0，然后设置边框样式

width: 0;

height: 0;

border-top: 40px solid transparent;

border-left: 40px solid transparent;

border-right: 40px solid transparent;

border-bottom: 40px solid #ff0000;

### 14. 一个满屏品字布局如何设计

第一种真正的品字：

三块高宽是确定的；

上面那块用margin: 0 auto;居中；

下面两块用float或者inline-block不换行；

用margin调整位置使他们居中。

第二种全屏的品字布局:

上面的div设置成100%，下面的div分别宽50%，然后使用float或者inline使其不换行。

### 15 常见的兼容性问题？

不同浏览器的标签默认的margin和padding不一样。\*{margin:0;padding:0;}

IE6双边距bug：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。hack：display:inline;将其转化为行内属性。

渐进识别的方式，从总体中逐渐排除局部。首先，巧妙的使用“9”这一标记，将IE浏览器从所有情况中分离出来。接着，再次使用“+”将IE8和IE7、IE6分离开来，这样IE8已经独立识别。

{

background-color:#f1ee18;/\*所有识别\*/

.background-color:#00deff\9; /\*IE6、7、8识别\*/

+background-color:#a200ff;/\*IE6、7识别\*/

\_background-color:#1e0bd1;/\*IE6识别\*/

}

设置较小高度标签（一般小于10px），在IE6，IE7中高度超出自己设置高度。hack：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。

IE下，可以使用获取常规属性的方法来获取自定义属性,也可以使用getAttribute()获取自定义属性；Firefox下，只能使用getAttribute()获取自定义属性。解决方法:统一通过getAttribute()获取自定义属性。

Chrome 中文界面下默认会将小于 12px 的文本强制按照 12px 显示,可通过加入 CSS 属性 -webkit-text-size-adjust: none; 解决。

超链接访问过后hover样式就不出现了，被点击访问过的超链接样式不再具有hover和active了。解决方法是改变CSS属性的排列顺序:L-V-H-A ( love hate ): a:link {} a:visited {} a:hover {} a:active {}

### 16 为什么要初始化CSS样式

因为浏览器的兼容问题，不同浏览器对有些标签的默认值是不同的，如果没对CSS初始化往往会出现浏览器之间的页面显示差异。

### 17 absolute的containing block计算方式跟正常流有什么不同？

无论属于哪种，都要先找到其祖先元素中最近的 position 值不为 static 的元素，然后再判断：

若此元素为 inline 元素，则 containing block 为能够包含这个元素生成的第一个和最后一个 inline box 的 padding box (除 margin, border 外的区域) 的最小矩形；

否则,则由这个祖先元素的 padding box 构成。

如果都找不到，则为 initial containing block。

补充：

static(默认的)/relative：简单说就是它的父元素的内容框（即去掉padding的部分）

absolute: 向上找最近的定位为absolute/relative的元素

fixed: 它的containing block一律为根元素(html/body)

### 18 CSS里的visibility属性有个collapse属性值？在不同浏览器下以后什么区别？

当一个元素的visibility属性被设置成collapse值后，对于一般的元素，它的表现跟hidden是一样的。

chrome中，使用collapse值和使用hidden没有区别。

firefox，opera和IE，使用collapse值和使用display：none没有什么区别。

### 19 display:none与visibility：hidden的区别？

display：none 不显示对应的元素，在文档布局中不再分配空间（回流+重绘）

visibility：hidden 隐藏对应元素，在文档布局中仍保留原来的空间（重绘）

### 20 position跟display、overflow、float这些特性相互叠加后会怎么样？

display属性规定元素应该生成的框的类型；position属性规定元素的定位类型；float属性是一种布局方式，定义元素在哪个方向浮动。

类似于优先级机制：position：absolute/fixed优先级最高，有他们在时，float不起作用，display值需要调整。float 或者absolute定位的元素，只能是块元素或表格。

### 21 对BFC规范(块级格式化上下文：block formatting context)的理解？

BFC规定了内部的Block Box如何布局。

定位方案：

内部的Box会在垂直方向上一个接一个放置。

Box垂直方向的距离由margin决定，属于同一个BFC的两个相邻Box的margin会发生重叠。

每个元素的margin box 的左边，与包含块border box的左边相接触。

BFC的区域不会与float box重叠。

BFC是页面上的一个隔离的独立容器，容器里面的子元素不会影响到外面的元素。

计算BFC的高度时，浮动元素也会参与计算。

满足下列条件之一就可触发BFC

根元素，即html

float的值不为none（默认）

overflow的值不为visible（默认）

display的值为inline-block、table-cell、table-caption

position的值为absolute或fixed

### 22 为什么会出现浮动和什么时候需要清除浮动？清除浮动的方式？

浮动元素碰到包含它的边框或者浮动元素的边框停留。由于浮动元素不在文档流中，所以文档流的块框表现得就像浮动框不存在一样。浮动元素会漂浮在文档流的块框上。

浮动带来的问题：

父元素的高度无法被撑开，影响与父元素同级的元素

与浮动元素同级的非浮动元素（内联元素）会跟随其后

若非第一个元素浮动，则该元素之前的元素也需要浮动，否则会影响页面显示的结构。

清除浮动的方式：

父级div定义height

最后一个浮动元素后加空div标签 并添加样式clear:both。

包含浮动元素的父标签添加样式overflow为hidden或auto。

父级div定义zoom

### 23 上下margin重合的问题

在重合元素外包裹一层容器，并触发该容器生成一个BFC。

例子：

<div class="aside"></div>

<div class="text">

<div class="main"></div>

</div>

<!--下面是css代码-->

.aside {

margin-bottom: 100px;

width: 100px;

height: 150px;

background: #f66;

}

.main {

margin-top: 100px;

height: 200px;

background: #fcc;

}

.text{

/\*盒子main的外面包一个div，通过改变此div的属性使两个盒子分属于两个不同的BFC，以此来阻止margin重叠\*/

overflow: hidden; //此时已经触发了BFC属性。

}

### 24设置元素浮动后，该元素的display值是多少？

自动变成display:block

### 25 移动端的布局用过媒体查询吗？

通过媒体查询可以为不同大小和尺寸的媒体定义不同的css，适应相应的设备的显示。

<head>里边<link rel=”stylesheet” type=”text/css” href=”xxx.css” media=”only screen and (max-device-width:480px)”>

CSS : @media only screen and (max-device-width:480px) {/css样式/}

### 26 使用 CSS 预处理器吗？

Less sass

### 27 CSS优化、提高性能的方法有哪些？

避免过度约束

避免后代选择符

避免链式选择符

使用紧凑的语法

避免不必要的命名空间

避免不必要的重复

最好使用表示语义的名字。一个好的类名应该是描述他是什么而不是像什么

避免！important，可以选择其他选择器

尽可能的精简规则，你可以合并不同类里的重复规则

### 28 浏览器是怎样解析CSS选择器的？

CSS选择器的解析是从右向左解析的。若从左向右的匹配，发现不符合规则，需要进行回溯，会损失很多性能。若从右向左匹配，先找到所有的最右节点，对于每一个节点，向上寻找其父节点直到找到根元素或满足条件的匹配规则，则结束这个分支的遍历。两种匹配规则的性能差别很大，是因为从右向左的匹配在第一步就筛选掉了大量的不符合条件的最右节点（叶子节点），而从左向右的匹配规则的性能都浪费在了失败的查找上面。

而在 CSS 解析完毕后，需要将解析的结果与 DOM Tree 的内容一起进行分析建立一棵 Render Tree，最终用来进行绘图。在建立 Render Tree 时（WebKit 中的「Attachment」过程），浏览器就要为每个 DOM Tree 中的元素根据 CSS 的解析结果（Style Rules）来确定生成怎样的 Render Tree。

### 29 在网页中的应该使用奇数还是偶数的字体？为什么呢？

使用偶数字体。偶数字号相对更容易和 web 设计的其他部分构成比例关系。Windows 自带的点阵宋体（中易宋体）从 Vista 开始只提供 12、14、16 px 这三个大小的点阵，而 13、15、17 px时用的是小一号的点。（即每个字占的空间大了 1 px，但点阵没变），于是略显稀疏。

### 30 margin和padding分别适合什么场景使用？

何时使用margin：

需要在border外侧添加空白

空白处不需要背景色

上下相连的两个盒子之间的空白，需要相互抵消时。

何时使用padding：

需要在border内侧添加空白

空白处需要背景颜色

上下相连的两个盒子的空白，希望为两者之和。

兼容性的问题：在IE5 IE6中，为float的盒子指定margin时，左侧的margin可能会变成两倍的宽度。通过改变padding或者指定盒子的display：inline解决。

### 31 元素竖向的百分比设定是相对于容器的高度吗？

当按百分比设定一个元素的宽度时，它是相对于父容器的宽度计算的，但是，对于一些表示竖向距离的属性，例如 padding-top , padding-bottom , margin-top , margin-bottom 等，当按百分比设定它们时，依据的也是父容器的宽度，而不是高度。

### 32 全屏滚动的原理是什么？用到了CSS的哪些属性？

原理：有点类似于轮播，整体的元素一直排列下去，假设有5个需要展示的全屏页面，那么高度是500%，只是展示100%，剩下的可以通过transform进行y轴定位，也可以通过margin-top实现

overflow：hidden；transition：all 1000ms ease；

### 33 什么是响应式设计？响应式设计的基本原理是什么？如何兼容低版本的IE？

响应式网站设计(Responsive Web design)是一个网站能够兼容多个终端，而不是为每一个终端做一个特定的版本。

基本原理是通过媒体查询检测不同的设备屏幕尺寸做处理。

页面头部必须有meta声明的viewport。

<meta name="’viewport’" content="”width=device-width," initial-scale="1." maximum-scale="1,user-scalable=no”"/>

### 34 视差滚动效果？

视差滚动（Parallax Scrolling）通过在网页向下滚动的时候，控制背景的移动速度比前景的移动速度慢来创建出令人惊叹的3D效果。

CSS3实现

优点：开发时间短、性能和开发效率比较好，缺点是不能兼容到低版本的浏览器

jQuery实现

通过控制不同层滚动速度，计算每一层的时间，控制滚动效果。

优点：能兼容到各个版本的，效果可控性好

缺点：开发起来对制作者要求高

插件实现方式

例如：parallax-scrolling，兼容性十分好

### 35 ::before 和 :after中双冒号和单冒号有什么区别？解释一下这2个伪元素的作用

单冒号(:)用于CSS3伪类，双冒号(::)用于CSS3伪元素。

::before就是以一个子元素的存在，定义在元素主体内容之前的一个伪元素。并不存在于dom之中，只存在在页面之中。

:before 和 :after 这两个伪元素，是在CSS2.1里新出现的。起初，伪元素的前缀使用的是单冒号语法，但随着Web的进化，在CSS3的规范里，伪元素的语法被修改成使用双冒号，成为::before ::after

### 36 你对line-height是如何理解的？

行高是指一行文字的高度，具体说是两行文字间基线的距离。CSS中起高度作用的是height和line-height，没有定义height属性，最终其表现作用一定是line-height。

单行文本垂直居中：把line-height值设置为height一样大小的值可以实现单行文字的垂直居中，其实也可以把height删除。

多行文本垂直居中：需要设置display属性为inline-block。

### 37 怎么让Chrome支持小于12px 的文字？

p{font-size:10px;-webkit-transform:scale(0.8);} //0.8是缩放比例

### 38 让页面里的字体变清晰，变细用CSS怎么做？

-webkit-font-smoothing在window系统下没有起作用，但是在IOS设备上起作用-webkit-font-smoothing：antialiased是最佳的，灰度平滑。

### 39 position:fixed;在android下无效怎么处理？

<meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0, user-scalable=no"/>

### 40 如果需要手动写动画，你认为最小时间间隔是多久，为什么？

多数显示器默认频率是60Hz，即1秒刷新60次，所以理论上最小间隔为1/60＊1000ms ＝ 16.7ms。

### 41 li与li之间有看不见的空白间隔是什么原因引起的？有什么解决办法？

行框的排列会受到中间空白（回车空格）等的影响，因为空格也属于字符,这些空白也会被应用样式，占据空间，所以会有间隔，把字符大小设为0，就没有空格了。

解决方法：

可以将<li>代码全部写在一排

浮动li中float：left

在ul中用font-size：0（谷歌不支持）；可以使用letter-space：-3px

### 42 display:inline-block 什么时候会显示间隙？

有空格时候会有间隙 解决：移除空格

margin正值的时候 解决：margin使用负值

使用font-size时候 解决：font-size:0、letter-spacing、word-spacing

### 43 有一个高度自适应的div，里面有两个div，一个高度100px，希望另一个填满剩下的高度

外层div使用position：relative；高度要求自适应的div使用position: absolute; top: 100px; bottom: 0; left: 0

### 44 png、jpg、gif 这些图片格式解释一下，分别什么时候用。有没有了解过webp？

png是便携式网络图片（Portable Network Graphics）是一种无损数据压缩位图文件格式.优点是：压缩比高，色彩好。 大多数地方都可以用。

jpg是一种针对相片使用的一种失真压缩方法，是一种破坏性的压缩，在色调及颜色平滑变化做的不错。在www上，被用来储存和传输照片的格式。

gif是一种位图文件格式，以8位色重现真色彩的图像。可以实现动画效果.

webp格式是谷歌在2010年推出的图片格式，压缩率只有jpg的2/3，大小比png小了45%。缺点是压缩的时间更久了，兼容性不好，目前谷歌和opera支持。

### 45 style标签写在body后与body前有什么区别？

页面加载自上而下 当然是先加载样式。

写在body标签后由于浏览器以逐行方式对HTML文档进行解析，当解析到写在尾部的样式表（外联或写在style标签）会导致浏览器停止之前的渲染，等待加载且解析样式表完成之后重新渲染，在windows的IE下可能会出现FOUC现象（即样式失效导致的页面闪烁问题）

### 46 CSS属性overflow属性定义溢出元素内容区的内容会如何处理?

参数是scroll时候，必会出现滚动条。

参数是auto时候，子元素内容大于父元素时出现滚动条。

参数是visible时候，溢出的内容出现在父元素之外。

参数是hidden时候，溢出隐藏。

### 47 阐述一下CSS Sprites

将一个页面涉及到的所有图片都包含到一张大图中去，然后利用CSS的 background-image，background- repeat，background-position 的组合进行背景定位。利用CSS Sprites能很好地减少网页的http请求，从而大大的提高页面的性能；CSS Sprites能减少图片的字节。

### 48. 移动端的左右布局和按钮等css样式实现

### 49. css有哪些新增的动画特性

## JS基础知识

### 1. js中的事件循环问题（Event loop）

Js的解析是由浏览器里面的js解析引擎完成的。Js是单线程运行，对于某些耗时的任务，为了避免堵塞页面，需要一个机制去先执行后面的任务，也就是同步任务和异步任务，js的执行机制就可以看做主线程上面加上一个任务队列。同步任务就是放在主线程上执行的任务，异步任务是放在任务队列里的任务。所有的同步任务在主线程执行，形成一个执行栈，异步任务有了执行结果就在任务队列里面放置一个事件；脚本运行时先一次运行执行栈，然后从任务队列里面提取事件，运行任务队列里的任务，这个过程是不断重复的，所以叫事件循环

### 2. 为什么js要放在最后加载？

浏览器在解析过程中，如果遇到请求外部资源时，请求过程是异步的，不影响html文档进行加载，但是当文档加载过程中遇到js文件，html文档就会挂起，直到js文件加载完毕并且解析执行完毕才会继续html的渲染过程。因为js有可能修改dom结构，所以js执行完成前后续文件的加载时没有必要的，这就是js阻塞后续资源下载的原因。Css文件的加载不影响js的加载，却影响js文件的执行，js代码执行前必须保证css文件已经下载并加载完毕

### 3. 什么是变量提升？

### 4. 什么是线程和进程

进程是cpu资源分配的最小单位（能用于资源和独立运行的最小单位）

线程是cpu调度的最小单位（线程是建立在进程基础上的一次程序运行单位，一个进程中可以有多个线程）

不同进程之间也可以进行通信，但是代价较大

单线程和多线程其实指的是一个进程内的单和多

### 5. 浏览器时多进程还是单进程的？包含了哪些进程？有什么优势

浏览器是多进程的，浏览器的主进程是broweser进程只有一个，负责浏览器界面的显示，与用户交互，并且负责各个页面的管理创建和销毁其他进程，在浏览器中打开一个网页相当于新起了一个进程，当然如果打开多个标签页就会合并成为一个进程

（这一部分单独作为一个知识点进行总结）

### 6. 事件绑定有哪几种方法，有哪些参数？

在元素中存在大量的事件绑定的时候，一般使用事件委托的方式来绑定事件

* 嵌入dom，直接在dom元素上面添加时间

<button onclick="open()">按钮</button>

<script>

function open(){

alert(1)

}

</script>

* 直接绑定，选择dom元素，绑定事件

<button id="btn">按钮</button>

<script>

document.getElementById('btn').onclick = function(){

alert(1)

}

</script>

* 事件监听

<button id="btn">按钮</button>

<script>

document.getElementById('btn').addEventListener('click',function(){

alert(1)

})

//兼容IE

document.getElementById('btn').attachEvent('click',function(){

alert(1)

})

</script>

### 7. 一个经典的问题：这段代码问题的根本原因是什么？怎么解决？

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

setTimeout(function(){

console.log(i);

},1000\*i);

}

一般情况下我们都知道这种情况的解决办法是闭包，但实际上它出现的根本原因是变量提升，console.log中i的值未定义，所以由于变量提升，i的值编程了for循环中的值，所以打印出来的值都是5，解决的办法可以使用闭包或者是let定义i

### 8. 关于promise你了解多少？里面有哪些参数？

可见Promise最大的好处是在异步执行的流程中，把执行代码和处理结果的代码清晰地分离了

Promise的工作原理：

Promise的回调函数不是传入的，而是使用then来调用的因此promise中定义的函数应该是马上执行的，then才是回调函数，放入queue队列中

同时在异步里面还有一个重要的概念：

* Macro-task: 包括script, setTimeout, setInterval, setImmediate,i/o, ui rendering
* Micro-task: 包括process.nextTick, Promise, Object.observe, MutationObserve

执行顺序：函数调用栈清空，只剩全局上下文，然后开始执行所有的micro-task，里面所有的任务执行完毕后，循环再次执行macro-task中的一个任务队列，执行完后在执行所有的micro-task ，这样一直循环

### 9. 面向对象里面的继承是怎么实现的

三种继承方法：（以两个对象Persion和Boy为例）

1. 原型继承：Boy.prototype = new Persion();

这种方法的优点是：既继承了父类的模板，又继承了父类的原型对象，缺点是父类实例传参，不是子类实例化传参

1. 类继承（借用构造函数的方式继承）Person.call(this,name,age);

这种方法的特点是：继承了父类的模板，不继承父类的原型的原型对象，优点是方便了子类实例传参，缺点是不继承父类的原型对象

1. 使用var Boy = object.Create(Person)来实现继承

实际使用第一种和第二种组合的方式来实现

### 10. js里面的this指向问题，注意要能了解到ES6/7里面的this变成了什么样

总的来说，this指向在函数定义的时候是确定不了的，只有函数执行的时候才能确定this到底指向谁，一般来说this指向的是最终调用它的对象。一般分下面的情况

1. 如果一个函数中有this，但是它并没有被上一级对象所调用，那么this指的是window，需要注意在严格模式下，this指向的是undefined

function a(){

var user = 'someOne';

console.log(this.user);//undefined

console.log(this);//window

}

a();//相当于window.a();

//this指向调用它的对象，这里的函数实际上是被window对象所调用

1. 如果函数中有this，这个函数被上一级的对象所调用，那么this指的是上一级对象

var o = {

user:'someOne',

fn: function(){

console.log(this.user);//someOne

console.log(this);//o对象

}

}

o.fn();//this被执行的时候指向 的是o

1. 如果一个函数中有this，这个函数包含多个对象，金冠这个函数被最外层的函数调用，this指向的也只是他上一级的对象

var o = {

a:10,

b:{

fn: function(){

console.log(this.a);//undefined

}

},

fn1: function(){

console.log(this.a);//10

}

}

o.fn1();//this指向的是o

o.b.fn();//this指向的是b

需要注意的是有两种特殊情况：

第一种是当this遇到return，如果返回值是一个对象，this指向这个返回的对象，如果不是一个对象，this还是指向函数的实例。需要注意null虽然是对象，但是this还是指向函数的实例

function fn(){

this.user = 'someOne';

return {};

}

var a = new fn();

console.log(a.user);//undefinded

function fn(){

this.user = 'someOne';

return function(){};

}

var a = new fn();

console.log(a.user);//undefinded

function fn(){

this.user = 'someOne';

return 1;

}

var a = new fn();

console.log(a.user);//someOne

function fn(){

this.user = 'someOne';

return undefined;

}

var a = new fn();

console.log(a.user);//someOne

function fn(){

this.user = 'someOne';

return null;

}

var a = new fn();

console.log(a.user);//someOne

第二种情况，虽然看着像是b调用的，实际上我们说的是最后调用的对象，这里最后执行的是j(),所以this指向的实际上是window

var o = {

a:10,

b:{

a:12,

fn:function(){

console.log(this.a)//undefined

console.log(this)//undefined

}

}

}

var j = o.b.fn;

j();

关于this实际上还可以改变this的指向，

1. New关键字可以改变this的指向，new的时候只是复制，执行的时候this指向的实际上是a

function fn(){

this.user = 'someOne';

}

var a = new fn();

console.log(a.user)//someOne

1. Call和apply，这时候this指向的是call里面的参数，注意如果参数是null，则this指向的是window

var a = {

user:"someOne",

fn:function(){

console.log(this.user);

}

}

var b = a.fn;

b.call(a);//call实际上把对象指向了a，所以打印出someOne

1. Bind函数，bind实际上返回的是一个函数

var a = {

user:"someOne",

fn:function(){

console.log(this.user);

}

}

var b = a.fn;

var c = b.bind(a);

console.log(c)//ƒ (){console.log(this.user);}

### 11. 作用域和作用域链问题

Js的作用域是面试时候永恒的主题，作用域和this，面向对象通常会被结合起来考察

其实问题分成五种类型，当然除此之外，有些面试官喜欢把ES6跟下面的五种结合起来考试，但原理是一样的。

1. Js作用域（全局变量，局部变量）内部可以访问外部，但是外部不能访问内部

var a=10;

function aaa(){

alert(a);

}

aaa();

//a为外部变量即全局变量，所以可以直接访问到，结果为10

function aaa(){

var a=10;

}

aaa();

alert(a);

//a为函数内部变量即局部变量，所以无法访问到，结果报错

var a=10;

function aaa(){

alert(a);

}

function bbb(){

var a=20;

aaa();

}

bbb();//结果为10，因为aaa函数不能访问到bbb()里面的局部变量，所以访问到的是a=10这个全局变量

1. 不用var定义变量时，会默认是全局变量(不规范，不推荐)

function aaa(){

a=10;

}

aaa();

alert(a);//结果为10

上面的函数等价于

var a;

function aaa(){

a=10;

}

aaa();

alert(a);

需要注意的是：给未声明的变量赋值，此变量就会变成全局变量；var a=b=10; 可以解析成b=10;var a = b;也就是说b是全局变量，a是局部变量，所以外部访问a访问不到，访问b的值为10.

function aaa(){

var a=b=10;

}

aaa();

alert(a);//结果为a未定义

alert(b);//结果为10

1. 变量查找的原则是就近原则去寻找定义的var变量，第二点是变量的声明被提前到作用域顶部，赋值保留在原地

function aaa(){

alert(a);

var a = 20;

}

aaa(); //结果是undefined

var a=10;

function aaa(){

alert(a);

var a=20;

}

aaa(); //结果是undefined，

这种情况要注意，上面的var a=10；其实是起迷惑作用的，可以把上面这个解析成下面的样式：

var a=10;

function aaa(){

var a;//局部作用域里面的声明被提前了，实际上并未赋值

alert(a);

var a=20;

}

aaa();

1. 当参数跟局部变量重名时，优先级是等同的。

var a=10;

function aaa(a){

alert(a);

var a = 20;//因为a是形参，优先级高于var a，所以局部变量a的声明其实被忽略了

}

aaa(a);//结果为10;

1. 变量修改的时候，另一个变量会跟着变化，但当变量被重新定义时，则另一个不变化

var a = [1,2,3];

var b=a;

b.push(4);

alert(a);//结果为[1,2,3,4]，当b改变的时候a也发生了改变

当b被重新赋值的时候，a不会改变，如：

var a=[1,2,3];

var b=a;

b=[1,2,3,4];

alert(a);//结果为[1,2,3,4]

### 12. js事件中的事件委托

### 13. 怎么把函数中的参数转化为数组

Var args = Array.prototype.slice.apply(arguments);

### 14. 关于函数柯里化的问题

函数柯里化就是把接收多个参数的函数转换为一个接受单一参数的函数，其实质上利用了闭包的概念。

## 浏览器知识

### 1. 浏览器是怎么编译js代码的？

### 2. chrome的v8引擎理解多少？

## HTTP知识：

### 1. 网络协议分几层？前端开发涉及到哪几层？

协议包括了7层：应用层，表示层，会话层，传输层，网络层，数据链路层，物理层

### 2. url的组成

### 3. 从页面上面输入URL到页面加载发生了什么？

* DNS解析，实际上就是从网址到IP地址的转换

这是个递归查询的过程，顺序是请求本地域名服务器—根域名服务器—com顶级域名服务器，然后查询到ip地址并缓存到本地

* TCP连接，tcp是http使用的传输层协议
* 发送HTTP请求
* 服务器处理请求并返回HTTP报文，

这部分主要有后端进行处理，对应的是编程语言中的socket，它对tcp连接进行处理，解析http协议报文，并且按照报文格式封装成为http request对象，供上层使用。这部分的工作由web服务器进行

* 浏览器解析渲染页面

浏览器把接收到的数据渲染到显示器上，首先解析html文件构建dom树，然后解析css文件构建渲染树，然后浏览器开始渲染树并将其绘制到屏幕上。这里面涉及到一个重排和重绘的问题

* 连接结束

### 3. 怎么去优化DNS？DNS缓存和DNS负载均衡

请求ip地址的时候经历了很多步骤，这样我们可以采用DNS缓存的方法去进行优化

DNS存在多级缓存，按照离浏览器的顺序依次分为浏览器缓存，系统缓存，路由器缓存，IPS服务器缓存，根域名服务器缓存，顶级域名服务器缓存，主域名服务器缓存

浏览器缓存在chrome://dns里面，系统缓存在电脑etc/hosts里面

Dns的缓存时间即为设置的ttl时间

Dns负载均衡就是根据每台服务器的负载量，离用户的远近去返回一个合适的ip给用户，也就是dns重定向，我们实际上用到的CDN就是dns重定向技术

### 4. webpack生产环境里面的ip地址是怎么请求的？

### 5. http的请求过程？request line,status line,head,body是什么样的

http请求过程就是构建http请求报文并通过tcp协议发送到服务器指定端口（http协议80/8080，https协议443）。http请求报文由三部分组成：请求行，请求报头和请求正文

请求行格式：Method Request-URL HTTP-VersionCRLF

常用的请求方式有：get，post，put，delete，options，head

请求报头：请求报头允许客户端向服务器传递请求的附加信息和客户端自身的信息，（不止是浏览器，还包括一些http测试工具，url等），常用的请求报头有：Accept(用于指定客户端接收哪些类型的信息)，Accept-Charset，Accept-Encoding，（指定接收的编码方式）Accept-Language,Content-Type,Authorization,Cookie,User-Agent，connection如果设置为keep-alive用于告诉客户端本次http请求使用相同的tcp通道，节省了tcp连接建立的时间

http响应报文也是有三部分组成：状态码，响应报头，响应报文

### 6. http常见的状态码有哪些？

状态码由3位数组成，第一个数字定义了响应的级别，有五种取值

1xx：指示信息-表示请求已接收，继续处理

2xx：成功，表示请求已被成功接收，理解，接收

3xx：重定向-要完成请求必须进一步的操作

4xx：客户端错误-请求有语法错误或请求无法实现

5xx：服务端错误-服务器未能实现合法的请求

常见的状态码有：200(请求成功)，204(请求没有内容返回)，304(不包含任何响应主体)，404(服务器找不到资源)，500(服务器错误)

### 7. request时候的get和post有什么区别？get时候的url encode是做什么用的？post里面的body有哪些类型（content-type）？

根据w3cSchool，get和post的区别在于：

* 后退和刷新按钮的时候，get是无害的，post的时候数据会被重新提交（浏览器应该告知用户）
* 书签，get可收藏为书签，post不能被收藏为书签
* 缓存，get能被缓存，post不能被缓存
* 编码类型，get的为application/x-www-form-urlencoded, post的为application/x-www-form-urlencoded或multipart/form-data.为二进制数据使用多重编码
* 历史，get的参数保留在浏览器历史中，post的参数不会保存在浏览器的历史中
* 对数据长度的限制，get受限制，当发送数据的时候get方法向url添加数据，url的长度是受限制的（url的最大长度是2048个字符），而post是无限制的
* 对数据类型的限制，get只允许ASCII字符，而post无限制，他允许二进制数据
* 安全性，相比之下get的安全性较差，因为所发送的数据是url的一部分，post相对安全，因为参数不会保存在浏览器历史或web服务器日志中
* 可见性，get的数据在url中对所有人可见，而post不会显示在url里面

### 8. head里面常用的有哪些字段？如connection，content-type,cache-control..

### 9. 提到cache-control，说说文件的缓存和过期机制.

### 10. 说到connection，keep-alive有什么作用

### 11. 提到keep-alive，那http通道与socket tcp通道有什么区别？http怎么保存长连接，polling是什么

### 12. tcp里面的三次握手指的是什么？slow start是啥？tcp的header的前20个字节各代表什么，options又了解多少？

### 13. tcp header里面的第14个字节所代表的8个flag是什么？tcpdump抓包的时候怎么理清一个http请求哪里开始，哪里结束

### 14. tcpdump怎么使用，有哪些常用的参数

### 15. 抓包的时候遇见的https是什么？简述https的过程？PKI是什么

http报文是明文包裹在TCP报文中发送的，服务端接收到tcp报文会解包提出http报文，https实际上就是在进入tcp报文前对http做了一次加密，https的实质是http+ssl（或TLS），所以从协议层面来看，https位于http协议和tcp协议之间

https在传输数据前需要客户端与服务器进行一次握手(TCP/SSL握手)，在握手过程中确立双方加密传输数据的密码信息，TCL/SSL使用非对称加密，对称加密及hash等，相对于http安全性更高，但是时间上必然有所损耗

### 16. 如果没用过tcpdump，那了解curl吗？

### 17. http是明文协议，安全性没有办法保证，怎么让里面的数据更安全？一些常用的对称加密算法是什么，怎么用

## Vue知识部分

### 1. 框架和库的区别

框架（framework）:有自己的语法特点，都有对应的各个模块

库（library）：专注于一点

框架的好处：思维模式的转变：从操作dom的模式切换到以数据为主

* 提高代码的质量，开发速度
* 提高代码的复用率
* 降低模块之间的耦合度（高内聚低耦合）

### 2. vue框架的一个概述

这是一个渐进式的构建用户界面的js框架，他可以做小到表单处理，大到复杂数据操作比较频繁的单页面应用程序。他有比较方便的中文文档，容易上手，体积小，基于组件化的开发方式，代码的可读性提高了。同时他也可以通过丰富的指令扩展模板，通过各种各样的插件来增强功能

### 3. vue中有哪些常见的指令？

* {{}}： 将表达式结果插入到调用元素的innerHTML中，还可以将数据绑定到视图
* 循环指令v-for:  <any v-for="tmp in array"></any>
* 选择指令v-if:判断表达式执行的结果来决定是否将当前元素挂载到DOM树
* 事件绑定v-on:<any v-on:eventName="handleEvent"></any>
* 属性绑定v-bind:  <any v-bind:myProp="表达式"></any>
* 双向数据绑定v-model：将视图中（表单元素）用户操作的结果绑定到数据

### 4. 组件化是什么意思？

组件就是可以反复使用的，带有特定功能的视图。组件化就是将封装的组件拼接在一起，够成一个复杂的页面应用程序

   new Vue({

           components:{

               'my-footer':{

              template:''

            }

           }

       })

注意事项：组件的id和使用方式要遵循烤串式命名a-b-c

如果一个组件要渲染多个元素，将多个元素放在一个顶层标签中

全局组件可以在id为example的范围内的任何一个组件内部，直接调用可以；但是局部组件只能在父模板中直接调用

### 5. vue中的生命周期

四个阶段：

* Create准备工作（数据初始化）
* Mount挂载前后针对元素进行操作
* Update数据发生变化
* Destory清理工作（关闭定时器，集合情况）

### 6. vue中的路由

路由的本质是建立url与页面之间的映射关系

Spa单页面应用只有一个完整的页面，它在加载页面时不会加载整个页面，而只是更新某个指定容器中的内容

路由的工作原理：

* 解析地址栏：完整的页面地址，路由地址
* 根据路由地址从路由词典中找到真正要加载的页面
* 发起ajax请求，请求要加载的页面
* 向指定的容器中插入加载来的页面

### 7. v-model实现的原理

监听原生组件的事件,当获取到原生组件的值后把 值通过调用 $emit('input' ,data) 方法去触发 input事件

### 8. vue repeat里面的key值起什么作用

## React/redux知识部分

## ES5和ES6新语法vue

## 正则表达式

（1）过滤出字母的正则表达式

[^(A-Za-z)]

（2）过滤出数字的正则表达式

[^(0-9)]

（3）过滤出中文的正则表达式

[^(\\u4e00-\\u9fa5)]

（4）过滤出字母、数字和中文的正则表达式

[^(a-zA-Z0-9\\u4e00-\\u9fa5)]

## Webpack工具

1. Webpack打包原理

Webpack的核心概念有四个：输入，输出，loader，plugins

2. 如何写webpack插件

## Nodejs实际经验

### 1. 能简单介绍一下nodejs吗？他最主要有哪些特性？

Nodejs是一个javascript的运行环境，它依赖于chrome v8引擎进行代码解释，他的特点是事件驱动型，非阻塞i/0，他比较轻量，可伸缩，适用于实时数据交互应用。

他主要的优势在于他提供了对系统中瓶颈的新的解决方案，比如

* 高并发，并发连接的处理中，他通过异步事件驱动模型提供了新的方案，详细的说就是更改连接到服务器的方式，每个连接发射一个在nodejs引擎中运行的事件，放到事件队列中，而不是为每个连接生成一个新的OS线程
* 适合I/o阻塞型应用，比如说我们需要从多个数据源拉取数据，然后进行处理，对于比如php这样的语言来说，一般是串行获取数据，而nodejs遇到i/o事件的时候会创建一个线程去执行，主进程会继续往下执行，在i/o操作完成后发射一个事件，事件代理接收后继续执行后面的逻辑。Nodejs的异步机制比较自然

Nodejs的缺点在于：

1. 不适合cpu密集型应用，cpu密集型应用给nodejs的挑战主要是由于js单线程的原因，如果有长时间运行的计算（比如大循环），将会导致cpu时间片不能释放，使得后续i/o无法发起
2. 只支持单核cpu，不能充分利用cpu
3. 可靠性比较低，一旦代码某个环节崩溃，整个系统都会崩溃（原因在于单进程单线程）

解决方案：

1. Nigx反向代理，负载均衡，开多个进程，绑定多个端口
2. 开多个进程监听同一个端口，使用cluster模块

Nodejs适合的场景在于：

1. Restful API，这是nodejs最适合的常见，可以适合数万条连接，本身没有太多逻辑，只需要请求api组织数据进行返回即可，他本质上只是从数据库中查找一些值，并将它们组成一些响应
2. 统一web应用的ui层

对于mvc的架构，实际上是两个ui层，一个是在浏览器里面我们最终看到的，另一个是server端，负责生产和拼接页面。但是如果我们想要进行面向服务的架构的话，为了更好的实现前后端分离，我们把所有的业务逻辑封装成rest调用，这样后端就只需要在后端处理业务逻辑就行

1. 大量ajax的应用

比如个性化的应用，每个用户看到的页面不一样，缓存失效，需要经常发送ajax的请求，这种情况下nodejs比较有优势

### 2. 对restful接口了解哪些？

Rest，意为表述性状态传递，他是一种针对网络应用的设计和开发方式，可以降低开发的复杂性，提高系统的可伸缩性，Rest是设计风格而不是标准，

Rest架构风格最重要的架构约束包括6个：

* 客户端服务器（通信只能由客户端单方面发起，表现为请求响应模式）
* 无状态，通信的会话状态应该完全由客户端来维护
* 缓存：响应内容可以在通信链的某处被缓存，以改善网络的效率
* 统一接口：通信链的组件之间通过统一的接口相互通信，以提高交互的可见性
* 分层系统：通过限制组件的行为，将架构分解为若干等级的层
* 按需代码：支持通过下载并执行一些代码，对客户端的功能进行扩展

### 3. nodejs进程守护怎么做？

### 4. nodejs进程监控怎么去实现

### 5. nodejs各个进程之间怎么去交互

### 6. 怎么保证在宕机的情况下数据不丢失

## 移动端碰到的问题

## 1. 移动端中页面头部的配置

### 2. 滚动穿透问题

滚动穿透是指在移动端有fixed遮罩背景和弹出层时，在屏幕上滑动能够滑动背景下面的内容。

解决方案：

* 设overflow为hidden

.modal-open {

&, body {

overflow: hidden;

height: 100%

}

}

当弹出层弹出时，在html上面添加.modal-open，禁用html和body的滚动条，但是实际上会发现

1. 由于body和html滚动条都被禁用，弹出层后面的滚动位置会丢失，需要使用js来计算位置，在弹出是需要保持滚动位置
2. 页面的背景还是有滚动的效果

* Js之touchmove+preventDefault

modal.addEventListener('touchmove', function(e) {

e.preventDefault();

}, false);

通过阻止移动端的touchmove事件，但是实际上还是会发现弹出层需要滚动时也会被阻止

* Position；fixed

body.modal-open {

position: fixed;

width: 100%;

}

这种方式同样当弹出层弹出时滚动条会丢失，所以还需要使用js保存滚动条的位置，在关闭弹出层时将滚动位置还原，这种方式比较好的解决了移动端滚动的问题

var ModalHelper = (function(bodyCls) {

var scrollTop; // 在闭包中定义一个用来保存滚动位置的变量

return {

afterOpen: function() { //弹出之后记录保存滚动位置，并且给body添加.modal-open

scrollTop = document.scrollingElement.scrollTop;

document.body.classList.add(bodyCls);

document.body.style.top = -scrollTop + 'px';

},

beforeClose: function() { //关闭时将.modal-open移除并还原之前保存滚动位置

document.body.classList.remove(bodyCls);

document.scrollingElement.scrollTop = scrollTop;

}

};

})('modal-open');

### 3. 移动端输入框被键盘挡住的问题

由于所开发的页面内嵌在公司的一个app里面，有一个类似于聊天窗口的界面，测试的时候发现部分安卓机中，输入框被完全遮挡。这个没有一套完整的解决方法

方法一：能够解决大部分安卓上面的问题

if(/Android/.test(navigator.appVersion)) {

window.addEventListener("resize", function() {

if(document.activeElement.tagName=="INPUT" || document.activeElement.tagName=="TEXTAREA") {

document.activeElement.scrollIntoView();

}

})

}

通过在安卓机中监听窗口resize的时候，判断当前获得焦点的元素是否为输入框，再 调用该元素的scrollintoview(),即将该元素展示在当前窗口的可是区域。由于这个方法被各个浏览器支持，所以这个方法最常用。这段代码在微信或其他浏览器上都有效果，但是需要内嵌在本公司app上，这个问题一直没有解决。后面测试的时候才发现，app内置的浏览器在聚焦输入框弹出键盘的时候没有触发窗口的resize事件，后面借鉴了阿里的一个网页版的聊天界面，通过获取去输入框焦点将输入框定位到窗口略高于输入框的位置，在失去焦点键盘弹回的时候再恢复到底部，这种方法暴力解决了这个问题。这种方法存在的问题是：无法监听resize事件，而且键盘的高度也不固定。所以大概保持高度设置在屏幕一般偏上一点，保证大部分情况下的输入框在键盘上面显示

### 4. ios滚动不平滑的问题

在移动端尤其是在ios中，当滑动屏幕的时候会发现手指一拿开滚动就停止，这种用户体验很不好，有种滚动卡顿的问题，css3里面的-webkit-overflow-scrolling可以完美解决这个问题，该属性可以控制元素在移动设备上是否使用滚动回弹的效果

1: auto,使用普通滚动，当手指从触摸屏上移开的时候，滚动会立即停止

2，touch，使用具有回弹效果的滚动，当手指从触摸屏上移开，内容会继续保持一段时间的滚动效果，继续滚动的速度和持续的时间和滚动手势的强烈程度成正比，同时会创建一个新的堆栈上下文

# 笔试&面试问题（会根据笔试题目进行扩展和延伸）

### ****1. 浏览器兼容性问题，举几个例子？你碰到过什么兼容性问题？****

**2. 浏览器上面的重排和重绘问题，怎么优化？**

### 2. 浮动布局怎么写？怎么清理浮动？

可以借鉴bootstrap里面的浮动样式

### 4. vue页面的跨域

利用node+webpack+webpack-dev-server代理接口跨域，在开发环境下，由于vue渲染服务和代理接口服务是同一个，所以页面与代理接口之间不用跨域

module.exports = {

entry: {},

module: {},

...

devServer: {

historyApiFallback: true,

proxy: [{

context: '/login',

target: 'http://www.domain2.com:8080', // 代理跨域目标接口

changeOrigin: true,

secure: false, // 当代理某些https服务报错时用

cookieDomainRewrite: 'www.domain1.com' // 可以为false，表示不修改

}],

noInfo: true

}

}

### 5. 什么叫跨域？跨域有哪些方式

跨域问题是由浏览器的同源策略导致的，同源指的是“协议+域名+端口”三者相同

* 通过jsonp跨域实现，即动态创建script，通过请求静态资源的方法实现跨域，其缺点在于只能使用get方式

Jquery和vue里面都有jsonp的方法，原生jsonp示例：

<script>

var script = document.createElement('script');

script.type='text/javascript'

script.src = 'http://www.domain2:8080/login?user=admin&callback=onBack';

document.head.appendChild(script);

function onBack(res){

alert(JSON.scringify(res));

}

</script>

后端代码示例：

var querystring = require('querystring');

var http = require('http');

var server = http.createServer();

server.on('request', function(req, res) {

var params = qs.parse(req.url.split('?')[1]);

var fn = params.callback;

// jsonp返回设置

res.writeHead(200, { 'Content-Type': 'text/javascript' });

res.write(fn + '(' + JSON.stringify(params) + ')');

res.end();

});

server.listen('8080');

console.log('Server is running at port 8080...');

* Document.domain + iframe跨域，通过js强制设置两个页面的document.domain实现同域，缺点是仅限主域相同，子域不同的场景(document.domain属性返回下载当前文档的服务器域名)

1. 父窗口（<http://www/domain.com/a.html>）

<iframe src="http://child.domain.com/b.html"></iframe>

<script>

document.domain = 'domain.com';

var user = 'admin'

</script

1. 子窗口（http://child.domain.com/b.html）

<script>

document.domain = 'domain.com';

alert('get js data from parent' + window.parent.user);

</script>

* Location.Hash + iframe

实现原理：a与b跨域通信，通过中间页c来实现。页面间不同域使用iframe的location.hash传值，相同域之间直接js访问

1. a.html(<http://www.domain1.com/a.html>)

<iframe id="iframe" src="http://www.domain2.com/b.html" style="display:none;"></iframe>

<script>

var iframe = document.getElementById('iframe');

//向b.html传hash值，并开放给同域c.html回调方法

function onCallback(res){

alert('data from c.html' + res)

}

</script>

1. b.html(<http://www.domain2.com/b.html>)

<iframe id="iframe" src="http://www.domain1.com/c.html" style="display:none;"></iframe>

<script>

var iframe = document.getElementById('iframe');

//监听a传来的hash值，并传递给c

window.onhashchange = function(){

iframe.src = iframe.src + location.hash;

}

</script>

1. c.html(<http://www/domain1.com/c.html>)

<script>

window.onhashchange = function(){

// 监听b.html里面传过来的hash值，通过操作同域a.html的js回调将结果传回

window.parent.parent.onCallback('hello:'+location.hash.replace('#user=',''));

}

</script>

* Window.name+iframe跨域

Window.name属性比较独特，name值在不同的页面（甚至不同域名）加载后依旧存在，并且支持很长的name值（2M）

* PostMessage跨域

PostMessage是h5中的api，并且是位数不多的可以跨域操作的window属性之一，一般用来解决以下问题：

1. 页面与其打开的新窗口的数据传递
2. 多窗口之间的消息传递
3. 页面与嵌套的iframe消息传递
4. 以上三个场景的跨域数据传递

用法：postmessage（data,origin）方法接收两个参数

data： html5规范支持任意基本类型或可复制的对象，但部分浏览器只支持字符串，所以传参时最好用JSON.stringify()序列化。

origin： 协议+主机+端口号，也可以设置为"\*"，表示可以传递给任意窗口，如果要指定和当前窗口同源的话设置为"/"。

1. a.html(<http://www.domain1.com/a.html>)

<iframe id="iframe" src=<http://www.domain2.com/b.html> style="display:none;"></iframe>

<script>

var iframe = document.getElementById('iframe');

iframe.onload =function(){

var data = {name:'aym'}

iframe.contentWindow.postMessage(JSON.stringify(data),'http://www.domain2.com/b.html');

}

window.addEventListener('message', function(e){

alert('data from domain2'+e.data);

},false);

</script>

1. b.html(<http://www.domain2.com/b.html>)

<script>

//接收domain1的值

window.addEventListener('message', function(e){

alert('data from domain2' + e.data);

var data = JSON.parse(e.data);

if(data){

data.number = 16;

window.parent.postMessage(JSON.stringify(data), 'http://www.domain1.com');

}

}, false);

</script>

* 跨域资源共享（CORS），主流的跨域解决方案

普通跨域请求：只需要服务端设置access-control-allow-origin即可，前端无需设置。如果需要携带cookie则需要前后端都需要修改

前端设置：原生js

var xhr = new XMLHttpRequest(); // IE8/9需用window.XDomainRequest兼容

// 前端设置是否带cookie

xhr.withCredentials = true;

xhr.open('post', 'http://www.domain2.com:8080/login', true);

xhr.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');

xhr.send('user=admin');

xhr.onreadystatechange = function() {

if (xhr.readyState == 4 && xhr.status == 200) {

alert(xhr.responseText);

}

};

后台设置：nodejs

var http = require('http');

var server = http.createServer();

var qs = require('querystring');

server.on('request', function(req, res) {

var postData = '';

// 数据块接收中

req.addListener('data', function(chunk) {

postData += chunk;

});

// 数据接收完毕

req.addListener('end', function() {

postData = qs.parse(postData);

// 跨域后台设置

res.writeHead(200, {

'Access-Control-Allow-Credentials': 'true', // 后端允许发送Cookie

'Access-Control-Allow-Origin': 'http://www.domain1.com', // 允许访问的域（协议+域名+端口）

'Set-Cookie': 'l=a123456;Path=/;Domain=www.domain2.com;HttpOnly' // HttpOnly:脚本无法读取cookie

});

res.write(JSON.stringify(postData));

res.end();

});

});

server.listen('8080');

console.log('Server is running at port 8080...');

* Nginx代理跨域

原理：同源策略是浏览器的安全策略，不是http协议的一部分，服务器调用http接口只使用http协议，而不会允许脚本，所以就不涉及跨域问题

实现：nginx配置一个代理服务器，域名与domain1一致，反向代理访问domain2接口，然后可以顺便修改cookie中的domain信息，方便当前域cookie写入

* Nodejs中间件跨域

通过中间件，启动一个代理服务器，也可以通过设置cookieDomainRewrite参数修改响应头中的cookie域名，实现当前域名cookie的写入，方便接口的写入认证

* Websocket协议跨域

Websocket协议是h5的新协议，它实现浏览器与服务器的全双工通信，同时允许跨域。我们可以使用socket.Io使用websocket

### 6. 怎么设置viewport，怎么处理移动端1px被渲染成2px的问题

<meta name="viewport" content="width=device-width,initial-scale=1.0,minimum-scale=1.0,

maximum-scale=1.0,user-scalable=no">

对于移动端1px被渲染成2px的问题来源于设备像素比，屏幕基本单位是一个点 1个点PC端表示1px 而移动端2个点表示1px 所以移动端比pc端看起来粗。Ui设计本身是按照2倍来设计的。这个问题有两种方案：

* 局部处理：meta中的viewport属性，initial-scale设置为1，rem按照设计稿走，外加利用transform的scale（0.5）
* 全局处理：meta中的viewport属性，initial-scale设置为0.5，rem按照设计稿走即可

### 7. 怎么进行渲染优化（雅虎的34条军规）

1. 禁止使用iframe(阻塞父文档的onload事件),

（Iframe会阻塞主页面的onload事件;不利于seo;iframe共享连接池，而浏览器对相同域的连接有限制，会影响页面的并行加载。如果必须使用iframe，使用js动态给iframe添加src属性值，这样可以避开两个问题）

1. 图片预加载，将样式表放在顶部，脚本放在底部，加上时间戳
2. 禁止使用gif图片实现loadng效果（避免cpu消耗，提高渲染性能）
3. 使用css3代码代替js动画（尽可能避免重排重绘及回流）
4. 对于小图使用base64编码来减少网络请求，但是大图不要用，比较费cpu(小图标的优势在于：减少了http请求，避免文件跨域，修改及时生效)
5. 页面头部的<script></script>会阻塞页面（因为render进程中 的js线程和渲染线程是互斥的）
6. 页面中空的href和src会阻塞页面其他资源的加载（阻塞下载进程）
7. 网页gzip，cdn托管，data缓存，图片服务器
8. 前端模板js+数据，减少html标签导致的带宽浪费，前端使用变量保存ajax请求结果，每次操作本地变量，不用请求，减少请求次数
9. 用innerhtml代替dom操作，减少dom操作次数，优化js性能
10. 当需要设置的样式很多的时候使用classname，不直接操作style
11. 少用全局变量，缓存dom结点查找的结果，减少io读取操作
12. 避免使用css表达式
13. 避免页面的主体布局中使用table，table要等其中内容完全下载完后才能显示，比较慢

对于普通网站的优化思路就是，尽量向前端优化，减少数据库操作，减少磁盘io

向前端优化指的是，在不影响功能和体验的情况下，能在浏览器执行的就不在服务器执行，能在缓存服务器上直接返回的就不到应用服务器，程序能直接取到的就不要从外部取得，本机内能取得的数据不要远程取，内存能取到达就不到磁盘取，缓存中有的就不到数据库查询

减少数据库操作指，减少更新次数，缓存结果减少查询次数，将数据库执行的操作尽可能在程序中完成（如join查询）

减少磁盘io指的是尽量不使用文件系统作为缓存，减少读写文件次数等。

程序优化的基本是慢的部分，而不是换语言

8. 事件的各个阶段

捕获阶段—>目标阶段—>冒泡阶段。

所以addEventListener的第三个参数设置为true和false的区别就很明显

True表示该元素在事件的捕获阶段（由外往内传递时）响应事件

False表示该元素在时间的冒泡阶段(由内向外传递时)响应事件

### 9. let var const有什么区别

Let允许声明一个作用域被限制在块级中的变量，语句或表达式，不允许重复声明

Let绑定不受变量提升的约束，这意味着let声明不会被提升到当前，该变量处于从块开始到初始化处理的暂存死区

Var 声明变量的作用域限制在其声明位置的上下文中，而非声明变量总是全局的

由于声明变量总是在任意代码执行之前处理的，所以在代码中的任意位置声明变量等效于在代码头部声明

Const 声明创建一个值的只读引用（即指针）

Js的常用类型有：string,number,boolean,array,object,null,undefined,其中基本类型有undefined，null,boolean,number,string保存在栈中，复合类型array，object保存在堆里

基本数据当值发生改变时，其对应的指针也将发生改变，造成const声明基本数据类型时，再将其改变就会报错，但如果是改变复合类型里面的某个value值时还是可以正常使用

10. 为什么var可以重复声明

当我们执行代码时，我们可以简单的理解为变量分配一块内存，但在运行时编译器和引擎还会进行两项额外的操作：判断变量是否已经声明

* 首先编译器对代码进行分析拆解，从左至右遇见 var a，则编译器会询问作用域是否已经存在叫 a 的变量了，如果不存在，则招呼作用域声明一个新的变量 a，若已经存在，则忽略 var 继续向下编译，这时 a = 2被编译成可执行的代码供引擎使用。
* 引擎遇见 a=2时同样会询问在当前的作用域下是否有变量 a，若存在，则将 a赋值为 2（由于第一步编译器忽略了重复声明的var，且作用域中已经有 a，所以重复声明会发生值得覆盖而并不会报错）。若不存在，则顺着作用域链向上查找，若最终找到了变量 a则将其赋值 2，若没有找到，则招呼作用域声明一个变量 a并赋值为 2。 参考链接

### 10. 箭头函数

语法比函数表达式更短，类似于匿名函数，但是区别在于箭头函数的this是词法作用域，由上下文决定，即调用它的对象。这样就不用使用var that = this的hack写法了

### 11. 快速让一个数组乱序

var arr = [1,2,3,4,5,6,7,8,9,10];

arr.sort(function(){

return Math.random() - 0.5;

})

### 12. 字体 font-family,中文一般使用微软雅黑，宋体;英文一般用helvetica

body{ font-family: Microsoft Yahei,SimSun, Helventica}

### 13. 可能用到的meta标签

<!-- 设置缩放 -->

<meta name="viewport" content="width=device-width, initial-scale=1, user-scalable=no, minimal-ui" />

<!-- 可隐藏地址栏，仅针对IOS的Safari（注：IOS7.0版本以后，safari上已看不到效果） -->

<meta name="apple-mobile-web-app-capable" content="yes" />

<!-- 仅针对IOS的Safari顶端状态条的样式（可选default/black/black-translucent ） -->

<meta name="apple-mobile-web-app-status-bar-style" content="black" />

<!-- IOS中禁用将数字识别为电话号码/忽略Android平台中对邮箱地址的识别 -->

<meta name="format-detection"content="telephone=no, email=no" />

其他meta标签

<!-- 启用360浏览器的极速模式(webkit) -->

<meta name="renderer" content="webkit">

<!-- 避免IE使用兼容模式 -->

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<!-- 针对手持设备优化，主要是针对一些老的不识别viewport的浏览器，比如黑莓 -->

<meta name="HandheldFriendly" content="true">

<!-- 微软的老式浏览器 -->

<meta name="MobileOptimized" content="320">

<!-- uc强制竖屏 -->

<meta name="screen-orientation" content="portrait">

<!-- QQ强制竖屏 -->

<meta name="x5-orientation" content="portrait">

<!-- UC强制全屏 -->

<meta name="full-screen" content="yes">

<!-- QQ强制全屏 -->

<meta name="x5-fullscreen" content="true">

<!-- UC应用模式 -->

<meta name="browsermode" content="application">

<!-- QQ应用模式 -->

<meta name="x5-page-mode" content="app">

<!-- windows phone 点击无高光 -->

<meta name="msapplication-tap-highlight" content="no">

### 14. 消除css3中的transition闪屏（css3问题）

过渡动画（没有启动硬件加速的情况下会出现抖动现象）

解决方案：

1. 改变视角

.css {

-webkit-transform-style: preserve-3d;

-webkit-backface-visibility: hidden;

-webkit-perspective: 1000;

}

1. 启动硬件加速（常用的方式是：translate3d,translateZ,transform）

.css {

-webkit-transform: translate3d(0,0,0);

-moz-transform: translate3d(0,0,0);

-ms-transform: translate3d(0,0,0);

transform: translate3d(0,0,0);

}

弊端在于硬件加速导致cpu性能占用量过大，电池电量消耗加大，所有不能滥用

### 15. android 4.x bug

1. 三星galaxy s4中自带浏览器不支持border-radius缩写
2. 同时设置border-radius和背景色的时候，背景色会移除
3. 部分手机（如三星）,a连接支持鼠标:visited事件，即连接访问后文字变成紫色
4. Android无法同时播放多音频audio
5. Oppo的border-radius会失效

### 18. css实现单行文本溢出显示

Overflow：hidden; text-overflow:ellipsis;white-space:nowrap;

### 19. 实现多行文本溢出显示

1. 对于webkit浏览器和移动端

display: -webkit-box;//将对象做为弹性伸缩盒子模型显示

-webkit-box-orient: vertical;//设置或检索伸缩盒子对象的子元素排列方式

-webkit-line-clamp: 3;//限制在一个块元素显示的文本的行数

overflow: hidden;

1. 不兼容ie6-7-8，这个方法有个弊端，没有超出行的情况下也会出现省略号，需要跟js配合优化

div {

position: relative;

line-height: 20px;

max-height: 40px;//height需为line-height的整数倍

overflow: hidden;

}

div:after {

content: "..."; position: absolute; bottom: 0; right: 0; padding-left: 40px;

background: -webkit-linear-gradient(left, transparent, #fff 55%);

background: -o-linear-gradient(right, transparent, #fff 55%);

background: -moz-linear-gradient(right, transparent, #fff 55%);

background: linear-gradient(to right, transparent, #fff 55%);

}//加渐变背景可以避免文字只显示一半

### 20. 让图文不可复制

一般情况下这种设置就行了

-webkit-user-select: none;

-ms-user-select: none;

-moz-user-select: none;

-khtml-user-select: none;

user-select: none;

对于那些网页复制的文本都会自动加上来源说明的，大致做的思路是

1. 答案区域监听copy事件，并阻止该事件的默认行为，
2. 获取选中的内容（window.getselection），加上版权信息，然后设置到剪切板（clipboarddata.setData()）

### 23. 最快捷的数组求最大值法

Var arr = [1,5,1,7,5,9]

Math.max(…arr)(…运算符是将一个数组转化为用逗号分隔的参数序列)

### 24. restful请求中共有几种方式？一一列出，get和post有什么区别（详细说一下）

### 25. vue父子组件嵌套时，组件内部的各个声明周期钩子触发先后顺序

我们可以把子组件当成function函数来看待，当父组件import子组件的时候就当做声明并加载了这个函数，在调用的时候才会去执行。各个声明周期钩子触发的先后顺序是：先创建父组件，然后创建子组件，当子组件创建完成并且实体dom挂载完成后父组件才挂载完成

### 26. 前端安全中XSS,CSFR攻击指的什么？如何避免

### 27. HTML5新增了哪些内容或api，使用过哪些

### 28. input和textarea的区别

### 29. 使用一个div模拟textarea的实现

### 30. 左右布局：左边定宽，右边自适应，不少于三种方法

### 31. css3用过哪些新特性

### 32. BFC和IFC

### 33. 对栅格的理解

### 34. 水平居中有哪些实现方式

### 35. 1像素边框问题（在不同移动端中的显示问题）

### 36. 图片懒加载

### 37. 实现页面加载进度条

### 38. 事件委托和事件捕获有什么区别，怎么使用

### 39. 实现extend函数

### 40. 为什么会有跨域的问题，及跨域问题的解决方式

### 41. jsonp原理，postmessage原理

### 42. 实现拖拽功能，比如把5个兄弟结点中的最后一个结点拖拽到节点1和2之间

### 43. 动画：setTimeout何时执行，requestAnimationFrame的优点

### 44. 手写parseInt的实现：要求简单一些，把字符串型的数字转化为真正的数字即可，但不能使用js原生的字符串转数字的api比如number（）

### 45. 在编写分页器组件的时候，为了减少服务端查询次数，点击“下一页”怎样确保还有数据可以加载（请求数据不会为空）

### 46. ES6新增了哪些特性，使用过哪些

### 47. js模块化的实践

### 48. require.Js的实现原理（如果使用过webpack，进一步会问两种打包的异同及优缺点）

### 49. promise的实现原理，进一步问async，await是否使用过

### 50. 实现gulp的功能

### 51. 使用前端框架（angular/vue/react）带来 哪些好处，相对于使用jquery

### 52. vue双向绑定的实现

### 53. 单页面应用，如何实现其路由功能

### 54. 项目中使用过哪些优化方法

### 55. 输入一个url，enter后发生了什么

### 56. 页面的渲染过程

### 57. 优化中会提到缓存的问题，问，静态资源或接口如何做缓存优化

### 58. 页面中dom节点太多，会出现什么问题，如何优化？

### 59. 介绍一下你做的这个项目，进一步细问：整个项目有哪些模块，你主要负责哪些

### 60. 你在项目中的角色

### 61. 你在项目中最出彩的地方，还有那些可以改进

### 62. 项目中碰到过什么困难，怎么解决的

### 63. 如果你是项目负责人，任务是怎么分配的，有没有关注过团队成员的成长问题

### 64. 前端安全的问题：CSRF和CSS

### 65. 为什么选择做前端

### 66. 你希望进入一个什么样的团队

### 67. 你有什么想问我的吗

### 68. 你们项目用到了哪些技术，技术的实现原理是什么，项目框架是怎么搭建的，采取安全措施了吗

### 69. 说一下你了解的css盒子模型

### IE的怪异盒模型和标准浏览器的盒模型，然后可以通过box-sizing属性控制两种盒模型的变换。

### 70. 说一下box-sizing的应用场景

### 71. 说一下你了解的弹性flex布局

### 72. 说一下一个未知宽高元素怎么上下左右垂直居中

### 说了一下flex弹性布局的实现，说了一下兼容性，扯到了postcss的一些东西，然后说了一下常规的兼容性比较好的实现。

### 73. 说一下原型链，对象，构造函数之间的一些联系

### 74. dom事件绑定的几种方式

说了三种，然后说了一些冒泡，默认事件，以及DOM2，DOM3级的一些标准。

### 75. 有没有了解http2.0,websocket,https，说一下你的理解及你所了解的特性

### 76. webpack的入口文件怎么配置，多个入口怎么分割

这个自己就说了一下自己的理解，以及自己用node写的多入口怎么配置，然后面试官说不是多入口配置，然后我又说了一下自己的理解，然后这题就过了。

### 77. 我看到你的项目用到了Babel的一个插件：transform-runtime以及stage-2，你说一下他们的作用

这个我也还算比较了解，就说了一下ES的一些API，比如generator啥的默认不转换，只转换语法，需要这个来转换，然后说profill啥的，扯了一下stage-1，stage-2，stage-3，这个问题回答还算清楚。

### 78. 我看到你的webpack配置用到webpack.optimize.UglifyJsPlugin这个插件，有没有觉得压缩速度很慢，有什么办法提升速度。

这个我主要回答了一下，我之前也没怎么了解，一个想到是缓存原理，压缩只重新压缩改变的，还有就是减少冗余的代码，压缩只用于生产阶段，然后面试官问还有呢？我就说，还可以从硬件上提升，可以得到质的飞跃，比如换台I9处理器的电脑。。。。

### 79. 了解http协议，说一下200和304的理解和区别

这个噼里啪啦说了一堆，协商缓存和强制缓存的区别，流程，还有一些细节，提到了expires,Cache-Control,If-none-match,Etag,last-Modified的匹配和特征，这一块之前有过比较详细的了解，所以还是应答如流。

### 80. . dom事件中target和currentTarget的区别

### 81. 说一下你平时怎么解决跨域的，以及后续jsonp原理和实现以及cors怎么设置

### 82. 说一下深拷贝的实现原理

这个也还好，就是考虑的细节不是很周全，先是说了一种 JSON.stringify 和 JSON.parse 的实现，以及这种实现的缺点，主要就是非标准JSOn格式无法拷贝以及兼容性问题，然后问了我有么有用过IE8的一个什么JSON框架，我也不记得是什么了，因为我压根没听过，然后说了一下尾递归实现深拷贝的原理，还问了我typeof null是啥，这个当然是Object。。。

### 83. 有没有自己写过webpack的loadr，他的原理是啥，loader怎么配置的

### 84. 有没有去研究webpack的一些原理和机制，怎么实现的

### 85.babel把ES6转换成ES5或ES3的原理是什么，有没有研究过

### 86. 关于git大型项目的团队合作，以及持续集成

### 87. 什么是函数柯里化？说一下js的api哪些永道了函数柯里化的实现？

这个我就说了一下函数柯里化一些了解，以及在函数式编程的应用，最后说了一下JS中bind函数和数组的reduce方法用到了函数柯里化

### 88. ES6的箭头函数this问题，以及扩展运算符

这一块主要是API和概念的问题，扯了一些规范以及严格模式下其他情况this只想问题。

### 89. js模块化common.js，umd，cmd规范的了解，以及ES6的模块化跟其他几种的区别，以及出现的意义

CommonJS, AMD, CMD是三章常见的模块加载规范，他们的区别在于使用 的场景不一样

* Commonjs适合服务端采用，nodejs使用的就是这个规范，
* AMD是一个异步加载模块规范，先把所有依赖都定义出来，然后在加载完成的回调函数里面执行，requireJS使用的这个模块
* CMD是seajs使用的规范，采用的是依赖就近，使用到这个模块的时候再去require，他与amd最大的区别是对依赖模块的处理不同，他们两个都是异步加载模块

Amd依赖前置，js可以知道依赖模块是谁，立刻加载，cmd是就近依赖，需要把模块解析后才知道依赖了哪些模块

### 90.说一下vue实现双向数据绑定的原理，以及vue.js和react.js异同点，如果让你选框架，你会怎么权衡这两个框架，分析一下

主要是发布订阅的设计模式，还有就是ES5的 Object.defineProperty 的getter和setter机制，然后顺便扯了一下Angular的脏检测，以及alvon.js最先用到这种方式。然后扯了一下vue.js和react.js异同点，权衡框架选择，调研分析之类，噼里啪啦说了一大堆。

### 91. 博客里面草稿的交互细节以及实现原理

### 92. 一个业务场景，面对产品的不断迭代，以及需求的变动该怎么应对，具体技术方案实现

具体业务场景，我就不一一描述，Boss在白板上画了一个大致的模块图，然后做了一些需求描述。

然后需求一层一层的改变，然后往下挖，主要是考察应对产品能力，以及对代码的可维护性和可拓展性这些考察，开放性问题，我觉得还考察一些沟通交流方面的能力，因为有些地方面试官故意说得很含糊，反正就是一个综合能力，以及对产品的理解，中间谈到怎么实现，也问到了一些具体的点，记得问到的有一下几个。

### 93. 业务场景：比如百度的一个服务不想让阿里使用，如果识别到是阿里的请求，然后跳转到404或者拒绝服务之类

主要是考察http协议头Referer，然后怎么判断是阿里的ip或者啥的，我也不太清楚。

### 94. 二分法查找的时间复杂度怎么求，是多少

### 95. xss是什么，攻击原理，怎么预防

这个很简单，跨站脚本攻击XSS(cross site scripting)，攻击类型主要有两种：反射型和存储型，简单说了一下如何防御：

转义

DOM解析白名单

第三方库

CSP 自己对web安全这块系统学习过，前前后后大约了解了很多，对于XSS,CSRF,点击劫持，Cookie安全，HTTP窃听篡改，密码安全，SQL注入，社会工程学都有一定了解，所以这个自然也不在话下。

### 96. 线性存储结构和链式存储结构有什么区别？以及优缺点

我是类比JS数组和对象来回答的，反正还算凑合吧，自己都数据结构这块多少还是有些印象，所以入了前端，对数据结构和算法确实一直淡忘了。

### 97. 分析一下移动端日历，pc端日历以及桌面日历的一些不同和需要注意的地方

这个我能想到的大致都说了一遍，不同的场景交互和细节以及功能都有所偏差，以及功能的侧重都可能不同。

### 98. 白板写代码，用最简洁的代码实现数组去重，你还能想到哪些

ES6实现：

[...new Set([1,2,3,1,'a',1,'a'])]

ES5实现：

[1,2,3,1,'a',1,'a'].filter(function(ele,index,array){

   return index===array.indexOf(ele)

})

### 99. 怎么实现草稿，多终端同步，以及冲突问题？

类比git的处理方式

### 100. 关于js对象的深拷贝和浅拷贝

浅拷贝只是变量指针的改变，并没有重新复制

深拷贝则是对象的各个属性复制，并且将源对象属性包含的对象也依次采用深复制的方法递归复制到新的对象上

但是所有深拷贝都避免不了的问题就是内存问题，（可能）

对象复制的方法（在hexo源码中刚看到的，适合于数组和对象，比较简单的问题可以解决）

function objCopy(obj) {

if (obj === null) {

return null;

} else if (Array.isArray(obj)) {

return obj.slice(); //数组的slice方法没有参数的话返回自身

} else {

var copy = {};

Object.keys(obj).forEach(function (k) {

copy[k] = obj[k];

});

return copy;

}

}

采用的方法可以使递归，也可以是js序列化（JSON.parse(JSON.stringify(obj))）

var cloneObj = function(obj){

var str, newobj = obj.constructor === Array ? [] : {};

if(typeof obj !== 'object'){

return;

} else if(window.JSON){

str = JSON.stringify(obj), //系列化对象

newobj = JSON.parse(str); //还原

} else {

for(var i in obj){

newobj[i] = typeof obj[i] === 'object' ?

cloneObj(obj[i]) : obj[i];

}

}

return newobj;

};

### 101.echarts图片下载什么原理

### 102.vue 和react钩子的生命周期

### 103,angular和vue底层的原理（双向数据绑定）

Angular是dirty check ，vue的原理是object.Defineprototype中的get和set方法

### 104. vuex是什么

### 105. 怎么判断字符串（比如var a = ‘abc’和var a= new Object(‘abc’)）

Object.prototype.toString.call(obj) == “[object String]”;

### 106. ssr是什么？

服务器端渲染

### 107. css3 的模糊属性

Filter:blur;

### 108. 关于es6里面的keys你了解所示

### 109.nodejs中的buffer是什么，能简单介绍一下nodejs吗

### 110.能说一下啥是微服务吗

微服务是一种架构风格，一个大型复杂软件应用由一个或多个微服务组成。系统中的各个微服务可被独立部署，各个微服务之间是松耦合的。每个微服务仅关注于完成一件任务并很好地完成该任务。在所有情况下，每个任务代表着一个小的业务能力。比较好的实现了软件开发中的低耦合高内聚。

### 111.什么是restful？

这是一种设计风格

### 112. 怎么获取一个元素到视图顶部的距离。

### 113. getBoundingClientRect获取的top和offsetTop获取的top区别

### 114. css问题；使用css样式显示下面的图形



思路一：把这个圆形看成矩形，使用伪类元素:after和:before在右上角和左下角增加两个背景色为黑色的小矩形，然后使用border-radius：50%来使大矩形变成圆形

思路二：将圆形旋转45度，恢复成矩形，然后使用border颜色的变化，把border-top和border-bottom变成黑色，然后使用border-radius:50%来变成圆形，最后使用css3的属性，trantform: rotate(45deg)旋转一下即可

### 115. 获取数组中的最大值

先排序后找出最大值：arr.sort.reverse()[0];或者是arr.sort(function(a,b){return b-a})

ES5的方法 Math.max.apply(null, [-1,1,1001,-52])

ES6的方法 Math.max(…[-1,1,1001,-52]);

### 116.对数组中的元素去重

ES5中的方法：

R= arr.filter(function (element, index, self){

Return self.indexOf(element) === index;

});

ES6的方法：

new Set(…[1,1,12,3,4])

### 117. 怎么判断是否是数组，字符串

判断是否是数组：

Object.prototype.toString.call(arr) == ‘[object Array]’;

ES5的方法： Array.isArray(arr);

判断是否是字符串：(包含两种情况：str = “abc”;str =new String(‘abc’); )

Object.prototype.toString.call(str) == ‘[object String]’;

### 118. 写一个函数clone，实现对数组，字符串，数字，对象，布尔值五种数据类型的深拷贝

一个讨巧的方法是JSON.parse(JSON.stringify(arg)), 这种方法只能处理哪些可以被json表示的对象，比如String,Number,Boolean,Array和扁平对象

比较常见的简介的深拷贝的方法可以使用jquery的extend方法：（参数里面加个true就能实现深拷贝）

### 119. 有一个按行存储的二层数组，写一个函数：1.返回的html使用表格渲染整个数组，2. 同时给每一列的最大值突出显示3. 不要使用lodash，jquery等



### 120. 写一个函数，将str = “abc”重复n次

<javascript框架设计>中写了8种repeat方法，来作为对字符串方法的扩展

function repeat(target, n){

return (new Array(n+1)).join(target);

}

### 121. 假设有一个超超超大型的Web项目，JS源代码压缩之后超过10MB，每次更新之后都需要用户重新加载JS是不可接受的，那么怎么样从工程的角度解决这种问题？

“增量式更新”的解决方案，简单地说就是在版本更新的时候不需要重新加载资源，只需要加载一段很小的 diff 信息，然后合并到当前资源上，类似 git merge 的效果。

1、用户端使用 LocalStorage 或者其它储存方案，存储一份原始代码+时间戳：

　　{

　　timeStamp: "20161026xxxxxx",

　　data: "aaabbbccc"

　　}

　　2、每次加载资源的时候向服务器发送这个时间戳；

　　3、服务器从接受到时间戳中识别出客户端的版本，和最新的版本做一次 diff，返回两者的 diff 信息：

　　diff("aaabbbccc","aaagggccc");

　　// 假设我们的diff信息这样表示：

　　// [3, "-3", "+ggg", 3]

　　4、客户端接收到这个 diff 信息之后，把本地资源和时间戳更新到最新，实现一次增量更新：

　　mergeDiff("aaabbbccc",[3,"-3","+ggg",3]);

//=> "aaagggccc"

### 122 不用require怎么实现异步加载

自动生成js的版本号

### 123. WebPack和Grunt以及Gulp相比有什么特性

其实Webpack和另外两个并没有太多的可比性，Gulp/Grunt是一种能够优化前端的开发流程的工具，而WebPack是一种模块化的解决方案，不过Webpack的优点使得Webpack在很多场景下可以替代Gulp/Grunt类的工具。

Grunt和Gulp的工作方式是：在一个配置文件中，指明对某些文件进行类似编译，组合，压缩等任务的具体步骤，工具之后可以自动替你完成这些任务。

Webpack的工作方式是：把你的项目当做一个整体，通过一个给定的主文件（如：index.js），Webpack将从这个文件开始找到你的项目的所有依赖文件，使用loaders处理它们，最后打包为一个（或多个）浏览器可识别的JavaScript文件。

### 124. promise的执行顺序是什么样的，跟settimeout相比谁先执行

Pomise比其他所有的异步函数都先执行

### 125. 关于对象继承有几种方式？他们的优缺点是什么

New

Object.Create

Person.call(this) 缺点不能继承原型上的属性

### 126. 怎么更新浏览器上面的js缓存

在js文件后加时间戳

### 127. alert(1&&2),alert(1||0)

alert(1&&2)的结果是2

只要“&&”前面是false，无论“&&”后面是true还是false，结果都将返“&&”前面的值;

只要“&&”前面是true，无论“&&”后面是true还是false，结果都将返“&&”后面的值;

alert(0||1)的结果是1

只要“||”前面为false,不管“||”后面是true还是false，都返回“||”后面的值。

只要“||”前面为true,不管“||”后面是true还是false，都返回“||”前面的值。

### 128. mouseenter和mouseover的区别

* 不论鼠标指针穿过被选元素或其子元素，都会触发mouseover事件，对应mouseout
* 只有在鼠标指针穿过被选元素时，才会触发mouseenter事件，对应mouseleave

### 129. 用正则表达式匹配字符串，以字母开头，后面是数字，字符串或者下划线，长度为9-20

Var reg = /^[a-zA-Z][a-zA-Z0-9\_]{9,20}$/;

### 130. js字符串两边截取空白的trim的原型的方法的实现

 String.prototype.trim = **function** () {  
    **return** **this**.replace(/(^\s\*)|(\s\*$)/g,'');  
  };

### 131. 写出position不同值和区别

### 132. 怎样判断一个对象是否为空对象

1. 将json转化为json字符串，判断该字符串是否为{}

JSON.stringigy(obj) == ‘{}’

1. For in循环来进行判断，判断对象里面的属性值是否存在
2. 使用jquery里面的$isEmptyObject()方法
3. Object.getOwnPropertyNames()方法，获取对象中的属性，返回值是一个数组，通过判断数组的length来判断对象是否为空，不兼容IE8
4. 用ES6的Object.keys()方法，这个返回值也是个数组，判断数组的长度可以知道是否为空对象

# 前端代码题

### 1. js判断设备来源

function deviceType(){

var ua = navigator.userAgent;

var agent = ["Android", "iPhone", "SymbianOS", "Windows Phone", "iPad", "iPod"];

for(var i=0; i<len,len = agent.length; i++){

if(ua.indexOf(agent[i])>0){

break;

}

}

}

deviceType();

window.addEventListener('resize', function(){

deviceType();

})

微信的 有些不太一样

function isWeixin(){

var ua = navigator.userAgent.toLowerCase();

if(ua.match(/MicroMessenger/i)=='micromessenger'){

return true;

}else{

return false;

}

}

### 2. audio元素和video元素在ios和andriod中无法自动播放

原因： 因为各大浏览器都为了节省流量，做出了优化，在用户没有行为动作时（交互）不予许自动播放；

/音频，写法一

<audio src="music/bg.mp3" autoplay loop controls>你的浏览器还不支持哦</audio>

//音频，写法二

<audio controls="controls">

<source src="music/bg.ogg" type="audio/ogg"></source>

<source src="music/bg.mp3" type="audio/mpeg"></source>

优先播放音乐bg.ogg，不支持在播放bg.mp3

</audio>

//JS绑定自动播放（操作window时，播放音乐）

$(window).one('touchstart', function(){

music.play();

})

//微信下兼容处理

document.addEventListener("WeixinJSBridgeReady", function () {

music.play();

}, false);

//小结

//1.audio元素的autoplay属性在IOS及Android上无法使用，在PC端正常；

//2.audio元素没有设置controls时，在IOS及Android会占据空间大小，而在PC端Chrome是不会占据任何空间；

//3.注意不要遗漏微信的兼容处理需要引用微信JS；

### 3. 使用css实现一个持续的动画效果

考察css3中的animation的使用

animation: mymove 5s infinite;

@keyframes mymove(

from (top:0px;)

to (top:200px;)

)

### 4. 使用js实现一个持续的动画效果

可以使用定时器来实现，但是使用requestAnimationFrame更好

这个api的优势在于：

* 浏览器可以优化并行的动画动作，更合理的重新排列动作序列，并把能合并的动作放在一个渲染周期内完成，从而呈现更流畅的动画效果
* 解决毫秒的不精确性
* 避免过度渲染（渲染频率太高，tab不可见暂停等）

注：requestAnimationFrame和定时器一样也有一个类似的清除方法cancelAnimationFrame

//兼容性处理

window.requestAnimFrame = (function(){

return window.requestAnimationFrame ||

window.webkitRequestAnimationFrame ||

window.mozRequestAnimationFrame ||

function(callback){

window.setTimeout(callback, 1000/60);

}

})();

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

var flag = true;

var left = 0;

function render(){

left == 0? flag =true:left ==100?flag = false :'';

flag ? e.style.left = '$(left++)px':e.style.left = '$(left--)px';

}

(function animloop(){

render();

requestAnimframe(animloop);

})();

### 5. 前端有哪些布局？移动端的布局怎么写？

* 右边宽度固定，左边自适应

1. 弹性布局

<style>

body{

display: flex;

}

.left{

background-color:rebeccapurple;

height: 200px;

flex: 1;

}

.right{

background-color: red;

height:200px;

width: 100px;

}

</style>

<body>

<div class="left"></div>

<div class="right"></div>

</body>

1. 浮动布局

<style>

div{

height:200px;

}

.left{

float:right;

width:200px;

background-color: rebeccapurple;

}

.right{

margin-right:200px;

background-color: red;

}

</style>

<body>

<div class="left"></div>

<div class="right"></div>

</body>

* 水平垂直居中

1. 第一种：css3

margin-left : 50% ; transform: translateX(-50%);

1. 使用绝对定位，（可以使用margin调节位置也可以使用css3）

<style>

#container{

position:relative;

}

#center{

width: 100px;

height:100px;

position:absolute;

top: 50%;

left:50%;

transform: translate(-50%, -50%);

}

</style>

<body id="container">

<div id="center"></div>

</body>

1. Flex布局

<style>

#container{

display: flex;

justify-content: center;

align-items:center;

}

</style>

<body id="container">

<div id="center"></div>

</body>

* 垂直水平居中

提供四种方法：

1. 定位，盒子的宽高已知

position:absolute;

left:50%;

top:50%;

margin-left:width/2;

margin-top:height/2

1. Table-cell布局：

父级：display:table-cell;vertical-align:middle;

子级：margin:0 auto

1. 定位加transform，适用于子盒子 宽高不定（这里开启了硬件加速）

position:relative/absolute;

top:50%;

left:50%;

transform:translate(-50%,-50%)

1. Flex布局

父级：

display: flex;/\*flex 布局\*/

align-items: center; /\*实现垂直居中\*/

justify-content: center; /\*实现水平居中\*/

### 6 .写出下面函数的运行结果（promise异步）

setTimeout(function(){

console.log(1);

}, 0)

new Promise(function executor(resolve){

console.log(2);

for(var i = 0; i < 1000; i++){

i = 9999 && resolve();

}

console.log(3);

}).then(function(){

console.log(4);

})

console.log(5);

答案是2,3,5,4,1

首先执行setTimeout的时候把事件放到了任务队列里面，然后执行依次promise函数，执行promise后，执行正常顺序的函数，然后才执行异步函数。Promise的异步函数在所有其他异步函数之前执行，所以最后执行setTimeout

### 7. 写出下面代码的运行结果

var a = 10;

function test(){

a = 100;

console.log(a);

console.log(this.a);

var a;

console.log(a);

}

test();

答案： 100,10,100. Javascript在执行前会对整个脚本文件的声明部分做完整分析，包括局部变量，从而确定变量的作用域。所以在函数test执行前，由于第6行声明了局部变量a，所以函数内部的a都指向已经声明的局部变量。所以第四行输出100，第5行输出this.a，函数内部的this指针指向的是函数的调用者，这里函数test被全局对象调用，所以this指针指向全局变量（window），一开始声明的全局变量a=10，即window.a =10，第7行输出结果为100，因为局部变量a已经被赋值100，所以直接输出局部变量的值。

### 8. 写出下面代码的运行结果。

var a = 100;

function test(){

console.log(a);

var a = 10;

console.log(a);

}

test();

答案： undefined，10，这个是作用域里面变量提升的问题，在知识点里面讲过

### 9. 已知下类Animal，要求设计一个Cat类继承自Animal，并实现如下功能

//Animal:

function Animal(){

this.name = "Animal";

this.showName = function(){

console.log(this.name);

};

}

// cat:

function cat(){

this.name = "Cat";

this.showName1 = function(){

console.log(this.name);

};

this.showName2 = function(){

console.log(this.name);

};

this.showName3 = function(){

console.log(this.\_\_super.name+"=>"+this.name)

}

}

代码运行：请完善cat部分相关代码，得到如下结果：

var cat = new Cat();

console.log(cat instaceof Animal); //得到true，判断Anima.prototype是否存在于cat的原型链上

cat.showName1();//得到Cat，需要读到cat中的name属性

cat.showName2();//得到Animal,需要读到Animal中的name属性

cat.showName3();//得到"Animal"=>"Cat"(需要同时读到Cat中的name和Animal中的name)

答案：实际上考的是js中的继承方式，需要注意的是使用call的时候this指向会转移，所以在showname3的方法里面要再用一次Cat.call(this)把对象重新指回自己

//Animal:

function Animal(){

this.name = "Animal";

this.showName = function(){

console.log(this.name);

};

}

// cat:

function Cat(){

this.name = "Cat";

this.\_\_super = Cat.prototype;

this.showName1 = function(){

console.log(this.name);

};

this.showName2 = function(){

Animal.call(this);

// Animal.bind(this);

console.log(this.name);

};

this.showName3 = function(){

Cat.call(this);

console.log(this.\_\_super.name+"=>"+this.name)

}

}

Cat.prototype = new Animal();

var cat = new Cat();

console.log(cat instanceof Animal); //得到true

cat.showName1();//得到Cat，需要读到cat中的name属性

cat.showName2();//得到Animal,需要读到Animal中的name属性

cat.showName3();//得到"Animal"=>"Cat"(需要同时读到Cat中的name和Animal中的name)

### 10. 已知如下数组，

var arr = [[1,2,2],[3, 4, 5, 5],[6, 7, 8, 9,[11,12,[12,13,[14]]]],10];

编写一个程序将数组扁平化并去除其中重复部分数据，最终得到一个升序且不重复的数组，如var res= [1,2,3,4,5,6,7,8,9,10,11,12,13,14]

答案：

var arr = [[1,2,2],[3, 4, 5, 5],[6, 7, 8, 9,[11,12,[12,13,[14]]]],10];

function dealArray(arr){

// 首先将数组扁平化,使用递归的方法

function flatten(arr){

return arr.reduce(function(prev, item){

return prev.concat(Array.isArray(item)?flatten(item):item)

}, []);

}

//数组去重

function unique(arr){

return Array.from(new Set(arr));

}

var flattenArray = flatten(arr);

var uniqueArray = unique(flattenArray);

//数组排序

return uniqueArray.sort(function(a,b){

return a-b;

});

}

dealArray(arr);

### 11 写一个函数将一个数组中连续排序和和最大的子数组提取出来，比如[1,2,7,9,3,4,5]中是[3,4,5]，注意是两个条件：连续排序的子数组和数组各元素之和最大的子数组

### 12. 写一个函数add(),实现如下效果

console.log(add(1)(2)(3)(4)());//10

console.log(add(1,2)(3,4)());//10

console.log(add(1,2)(3,4));//10

答案：可以有两种思路：闭包实现和函数柯里化实现

# 对于具体项目中问题描述

### 1. 移动端适配是怎么做的

使用媒体查询做的响应式布局，根据不同的屏幕宽度加载不同 的css

### 2. 封装一个函数，参数是定时器的时间，.then执行回调函数

function sleep(time){

return new Promise((resolve) => setTimeout(resolve, time));

}

### 3. 关于this指向的问题

obj = {

name: 'a',

getName : function () {

console.log(this.name);

}

}

var fn = obj.getName

obj.getName()

var fn2 = obj.getName()

fn()

fn2()

### 4. commonJS的require/exports和ES6中的import/export的区别

* Commonjs模块的重要特性是加载时执行，即脚本代码在require的时候就会全部执行。一旦某个模块被循环加载，就只输出已经执行部分，还未执行部分不会输出
* ES6模块是动态引用，如果使用import从一个模块加载变量，那些变量不会被缓存，而是成为一个指向被加载模块的引用，需要开发者自己保证取值的时候可以取到值
* Import/export最终都是编译成为require/exports来执行的
* Commonjs规范规定，每个模块内部，module变量代表当前模块。这个模块是一个对象，它的exports属性是对外的接口，加载某个模块其实是加载该模块的module.exports属性
* Export命令规定的是对外的接口，必须与模块内部的变量建立一一对应的关系

### 5. 使用addEventListener点击li弹出内容，并且动态添加li后有效（事件冒泡）

<ul id="ul">

<li>1</li>

<li>2</li>

<li>3</li>

<li>4</li>

<li>5</li>

</ul>

<script>

var eleUl = document.getElementById('ul');

eleUl.addEventListener('click', function(e){

if(e.target && e.target.nodeName.toUpperCase() == "LI"){

alert(e.target.innerHTML)

}

}, false);

</script>

### 6. 怎么判断两个对象相等？

7. 项目做过哪些性能优化

* 减少http请求数
* 减少dns查询
* 使用cdn
* 避免重定向
* 图片懒加载
* 减少dom元素数量
* 减少dom元素操作
* 使用外部js和css
* 压缩js，css，字体，图片等
* 优化css sprite
* 使用iconfont
* 尽量减少iframe的使用
* 避免图片src为空

### 8. 模块化开发是怎么做的

使用命名空间

### 9. 有没有使用过webpack

Vue项目中使用过，自己平时代码练习的时候使用过

### 10. vue router跳转和location.href有什么区别

Router是hash改变，location.Href是页面跳转，刷新页面

### 11. vue双向绑定实现原理，你能自己实现一下双向绑定吗？

通过object.defineProperty实现的

自己实现双向绑定：

<div id="app">

<input type="text" id="txt">

<p id="show-txt"></p>

</div>

<script>

var obj = {}

Object.defineProperty(obj, 'txt',{

get: function(){

return obj

},

set: function (newValue){

document.getElementById('txt').value = newValue

document.getElementById('show-txt').innerHTML = newValue;

}

})

document.addEventListener('keyup', function(e){

obj.txt = e.target.value

})

### 12. react和vue有什么区别

### 13. set和map数据结构

Es6提供了新的数据结构set，类似于set，但是成员值都是唯一的，没有重复的值

Es6提供了map数据结构，类似于对象，也是键值对组合，但是键的范围不限于字符串，各种类型的值都可以作为键，即object提供的是字符串-值的对应，map提供的是值-值的对应，是一种更完善的hash结构实现

### 14. weakmap和map的区别

* Weakmap结构和map结构基本类似，唯一的区别在于它只接受对象作为键名，不接受其他类型值为键名，而且键名指向的对象不计入垃圾回收机制
* Weakmap最大的好处是可以避免内存泄露，一个仅被weakmap作为key引用的对象，会被垃圾回收机制回收掉
* Weakmap用于和map类似的set，get，has，delete，clear方法，没有任何与迭代有关的属性和方法

### 15. 重排和重绘

部分渲染树需要重新分析且结点尺寸需要重新计算，这是重排

由于结点的几何属性发生改变或者由于样式发生改变，屏幕的部分内容需要更新，这被称为重绘

### 16. 什么情况下会触发重排和重绘

* 添加，删除，更新dom结点
* 通过display:none隐藏一个dom结点 触发重拍重绘
* 通过visbility:hidden隐藏一个dom结点，只触发重绘，因为没有集合变化
* 移动或给页面中的dom结点添加动画
* 添加一个样式表，调整样式属性
* 用户行为，例如调整窗口大小，改变自豪，或者滚动

### 17. 浏览器缓存

浏览器缓存分为强缓存和协商缓存。当客户端请求某个资源时获取缓存的流程如下：

* 先根据这个资源的一些http header判断他是否命中强缓存，如果命中，则直接从本地获取缓存资源，不会发送请求到服务器
* 当强缓存没有命中时，客户端会发送请求到服务器，服务器通过另一些request header验证这个资源是否命中协商缓存，称为http再验证，如果命中，服务器将请求返回但不返回资源，而是告诉客户端直接从缓存中获取，客户端收到返回后就会从缓存中获取资源；
* 强缓存和协商缓存共同之处在于，如果命中缓存，服务器都不会返回资源
* 区别在于：强缓存不会发送请求到服务器，但协商缓存会
* 当协商缓存也没命中时，服务器就会将资源发送到客户端
* 当ctrl+f5 强制刷新网页是直接从服务器加载，跳过强缓存和协商缓存
* 当f5刷新页面时会跳过强缓存，但会检查协商缓存

### 18. 强缓存

Expires（值为一个绝对时间的GMT格式的时间字符串，代表缓存资源的过期时间）

Cache-control:max-age（强缓存利用其max-age值来判断资源的最大生命周期，它的值为s）

### 19. 协商缓存

Last-modified （值为资源最后更新时间，随服务器response返回）

If-modified-since(通过比较两个时间来判断资源在两次请求期间是否有过修改，如果没有修改，则命中协商缓存)

Etag 表示资源内容的唯一标识，随服务器response返回dns

If-none-match(服务器通过比较请求头部的if-none-match与当前的etag是否一致来判断资源是否在两次请求中有过修改，如果没有修改则命中协商缓存)

### 20. 怎么进行调试的

### 21. 碰见了哪些问题

### 22. 项目怎么架构的

### 23. 你做了些什么工作

### 24. 你写过组件，能不能大致介绍一下思路

### 25. 你用过哪些常用框架，你读过他们的源码吗

## 前端争议性的基础问题

### 1. 代码后面要不要加分号？

这个问题争议性非常大，没有标准答案，只有各人的习惯，我参考的是<奇舞周刊>里面的文章<http://mp.weixin.qq.com/s/IrzqWuHMninsRizJk_dfAg> 《扯不完的 JS 分号问题》

在vue项目里面是是默认不加分号的，但是在其他的项目里面最好加分号，js的作者Brendan Eich创建的ASI机制实际上是个句法错误纠正的程式，只是让人更自由的去写代码，但并不是就可以不加分号了

# 非技术性问题（规划等）

### 1. 对于将来有哪些规划