# 总体要求：（记住带几张白纸和笔）

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

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

# 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();**

## ****关于CSS****

### 1. css（cascading style sheets级联样式表）中的选择器是什么，有哪些？

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

### 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. 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

### 6. 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

### 7. 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;

}



### 8. 如何中止web worker

W.terminate();

### 9. 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();



### 10. 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”

### 11. 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))

### 12. 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

### 13. 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

### 14. 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;}

### 15. 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

### 16. 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

### 17. 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

### 18. 我们如何使用websql



### 19. 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

### 20. 我们如何实现应用缓存



### 21. 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



### 22. 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



## 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

## 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）？

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

## ES5和ES6新语法

## Webpack工具

1. Webpack打包原理

2. 如何写webpack插件

## Nodejs实际经验

1. 对restful接口了解哪些？

## 后端java的一些初级问题

# 笔试问题

1. 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。

2. 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

3. 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}$/;

4. 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,'');  
  };

5. 写出position不同值和区别

# 面试问题

### ****1. 浏览器兼容性问题，举几个例子？****

**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

### 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）

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

# 前端中常用的代码

### 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\*/

# 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

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)

### 1. 怎么进行调试的

### 2. 碰见了哪些问题

### 3. 项目怎么架构的

### 4. 你做了些什么工作

# 非技术性问题（规划等）

1. 对于将来有哪些规划