# 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）

# 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. 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

## 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

## ES5和ES6新语法

## Webpack工具

## Nodejs实际经验

1. 对restful接口了解哪些？

## 后端java的一些初级问题

# 开发中具体典型问题

### ****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

### 3. 前端有哪些布局？移动端的布局怎么写？

### 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

### 10. 箭头函数

### 11. 快速让一个数组乱序

### 12. 字体 font-family

### 13. 可能用到的meta标签

### 14. 消除css3中的transition闪屏（css3问题）

### 15. android 4.x bug

### 16. js判断设备来源

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

### 18. css实现单行文本溢出显示

### 19. 实现多行文本溢出显示

### 20. 让图文不可复制

### 21. 盒子垂直水平居中

### 22. 改变placeholder的字体颜色大小

### 23. 最快捷的数组求最大值法

### 24. 数组去重写法

### 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

### 26. 前端安全中XSS,CSFR攻击指的什么？如何避免

# 对于具体项目中问题描述

### 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