**5. 样式的层级关系，选择器优先级，样式冲突，以及抽离样式模块怎么写，说出思路，有无实践经验**

1. 样式的层级关系：一个是权重，另一个就是共用样式和私用样式了，比如说两个ul，它们的子元素除了背景色之外都一样，那可以直接用li {}来定义相同的公用样式，用 .ul\_1 li {} ， .ul\_2 li {} 来定义不相同的样式。可以根据元素之间的差别来选择用哪种方法。推荐用多层级的方式书写css选择器。
2. 选择器优先级:（!important>）id选择器>class选择器（属性选择器/伪类选择器）>标签选择器（伪元素选择器） 同类选择符条件下层级越多的优先级越高。优先级就近原则，同权重情况下样式定义最近者为准。载入样式以最后载入的定位为准。

3. 样式冲突:是因为选择器的优先程度不一样，导致自己期望的样式属性没有显现出来。　一组嵌套选择器的实际特性可以计算出来。基本的，使用ID选择器的值是100，使用class选择器的值是10，每个html选择器的值是1。它们加起来就可以计算出特性的值。

　　p的特性是1（一个html选择器）

　　div p的特性是2（两个html选择器）

　　.tree的特性是10（1个class选择器）

　　div p.tree的特性是1+1+10=12，（两个html选择器，一个class选择器）

　　#baobab的特性是100（1个ID选择器）

　　body #content .alternative p的特性是112（两个html选择器，一个ID选择器，一个类选择器）

基本上，一个选择器越多特性，样式冲突的时候将显示它的样式。

4.抽离样式模块：默认样式单独拿出来，使用@import和link来引入。

**7. px和em和rem的区别，CSS中link 和@import的区别是?**

px是绝对单位，页面按精确像素展示。em相对于父节点字体的大小继承的，如父节点文字为15px，子节点1em = 15px来代替，rem是CSS3新加属性，chrome/firefox/IE9+支持。是自适应像素单位，他主要应用于移动端。

@import和link区别:

差别1：老祖宗的差别，link属于XHTML标签，而@import完全是css提供的一种方式。

link标签除了可以加载css外，还可以做很多其他的事情，比如定义RSS，定义rel连接属性等，@import只能加载CSS。

差别2：加载顺序的差别：当一个页面被夹在的时候（就是被浏览者浏览的时候），link引用的CSS会同时被加载，而@import引用的CSS会等到页面全部被下载完再加载。所以有时候浏览@import加载CSS的页面时会没有样式（就是闪烁），网速慢的时候还挺明显。

差别3：兼容性的差别。由于@import是CSS2.1提出的所以老的浏览器不支持，@import只有在IE5以上的才能识别，而link标签无此问题，完全兼容。

差别4：使用dom控制样式时的差别。当时用JavaScript控制dom去改变样式的时候，只能使用link标签，因为@import不是dom可以控制的（不支持）。

差别5（不推荐）：@import可以在css中再次引入其他样式表，比如创建一个主样式表，在主样式表中再引入其他的样式表，如：

　　@import “sub1.css”;

　　@import “sub2.css”;

　　sub1.css

　　———————-

　　p {color:red;}

　　sub2.css

　　———————-

　　.myclass {color:blue}

这样有利于修改和扩展。

但是：这样做有一个缺点，会对网站服务器产生过多的HTTP请求，以前是一个文件，而现在确实两个或更多的文件了，服务器压力增大，浏览量大的网站还是谨慎使用。

**8. 了解过flex吗?**

2009年W3C 提出了一种新的方案----Flex 布局，可以简便、完整、响应式地实现各种页面布局。目前，它已经得到了所有浏览器的支持，这意味着，现在就能很安全地使用这项功能。

**「 JavaScript 篇 」**

**JavaScript 基础**

**1. JavaScript 里有哪些数据类型，解释清楚 null 和 undefined，解释清楚原始数据类型和引用数据类型。比如讲一下 1 和 Number(1)的区别。**

有基本数据类型和复合数据类型。

null是一个表示"无"的对象，转为数值时为0；undefined是一个表示"无"的原始值，转为数值时为NaN。null表示"没有对象"，即该处不应该有值。undefined表示"缺少值"，就是此处应该有一个值，但是还没有定义。

3）原始数据类型可以通过typeOf进行检测，引用数据类型检测通过instanceOf;基本数据类型是指存放在栈中的简单数据段，数据大小确定，内存空间大小可以分配，它们是直接按值存放的，所以可以直接按值访问引用类型存在堆的概念;

引用类型是存放在堆内存中的对象，变量其实是保存的在栈内存中的一个指针（保存的是堆内存中的引用地址），这个指针指向堆内存。引用类型数据在栈内存中保存的实际上是对象在堆内存中的引用地址。通过这个引用地址可以快速查找到保存中堆内存中的对象

<https://www.cnblogs.com/c2016c/articles/9328725.html.>

4）两者的区别就是原始类型和包装对象的区别。

ps:什么是包装对象?

对象Number、String、Boolean分别对应数字、字符串、布尔值，可以通过这三个对象把原始类型的值变成（包装成）对象

1代表number类型的一个值，Number（1）是一个强制转为数字类型的过程。

**2. 将一下 prototype 是什么东西，原型链的理解，什么时候用 prototype**

1）prototype翻译过来就是指原型，每个对象上都有这个属性，用它可以实现继承或者改变对象原有的属性和方法。每创建一个函数，函数上都有一个属性为 prototype，它的值是一个对象。 这个对象的作用在于当使用函数创建实例的时候，那么这些实例都会共享原型上的属性和方法。

2）原型链： 在 JavaScript 中，每个对象都有一个指向它的原型（prototype）对象的内部链接（proto）。这个原型对象又有自己的原型，直到某个对象的原型为 null 为止（也就是不再有原型指向）。这种一级一级的链结构就称为原型链（prototype chain）。 当查找一个对象的属性时，JavaScript 会向上遍历原型链，直到找到给定名称的属性为止;到查找到达原型链的顶部（Object.prototype），仍然没有找到指定的属性，就会返回 undefined。

3）使用 prototype的好处是不会额外产生内存，所有实例化后的对象都会从原型上继承这个方法。也就是需要一个子类拥有父类的某些特性（同种特性可以覆盖），又可以添加自己的特性，而不会影响父类时候使用prototype。

Prototype通常用来解决一个问题：对象的创建比较耗费资源。比如，对象创建的时候需要访问数据库、需要读取外部文件、需要使用网络，这些都是比较耗费时间和内存的。如果可以用clone来解决，就方便多了。当需要创建很多 拥有相同属性的对象时候需要clone。

**3. 函数里的this什么含义，什么情况下，怎么用。**

1）情况一：纯粹的函数调用

这是函数的最通常用法，属于全局性调用，因此this就代表全局对象Global。

　　function test(){

　　　　this.x = 1;

　　　　alert(this.x);

　　}

test(); // 1

为了证明this就是全局对象，我对代码做一些改变：

　　var x = 1;

　　function test(){

　　　　alert(this.x);

　　}

　　test(); // 1

运行结果还是1。再变一下：

　　var x = 1;

　　function test(){

　　　　this.x = 0;

　　}

　　test();

　　alert(x); //0

2）情况二：作为对象方法的调用

函数还可以作为某个对象的方法调用，这时this就指这个上级对象。

　　function test(){

　　　　alert(this.x);

　　}

　　var o = {};

　　o.x = 1;

　　o.m = test;

　　o.m(); // 1

3） 情况三： 作为构造函数调用

所谓构造函数，就是通过这个函数生成一个新对象（object）。这时，this就指这个新对象。

　　function test(){

　　　　this.x = 1;

　　}

　　var o = new test();

　　alert(o.x); // 1

运行结果为1。为了表明这时this不是全局对象，对代码做一些改变：

　　var x = 2;

　　function test(){

　　　　this.x = 1;

　　}

　　var o = new test();

　　alert(x); //2

运行结果为2，表明全局变量x的值没变。

4）情况四： apply调用

apply()是函数对象的一个方法，它的作用是改变函数的调用对象，它的第一个参数就表示改变后的调用这个函数的对象。因此，this指的就是这第一个参数。

　　var x = 0;

　　function test(){

　　　　alert(this.x);

　　}

　　var o={};

　　o.x = 1;

　　o.m = test;

　　o.m.apply(); //0

apply()的参数为空时，默认调用全局对象。因此，这时的运行结果为0，证明this指的是全局对象。

如果把最后一行代码修改为

　　o.m.apply(o); //1

运行结果就变成了1，证明了这时this代表的是对象o。

**14. apply和 call 什么含义，什么区别?什么时候用。(我有篇文章 重点分析过)**

**call方法**:   
语法：call(thisObj，Object)  
定义：调用一个对象的一个方法，以另一个对象替换当前对象。  
说明：call 方法可以用来代替另一个对象调用一个方法。call 方法可将一个函数的对象上下文从初始的上下文改变为由 this指向。

**apply方法**：   
语法：apply(thisObj，[argArray])  
定义：应用某一对象的一个方法，用另一个对象替换当前对象。 对于apply和call两者在作用上是相同的，但两者在参数上有以下

**区别**：  
对于第一个参数意义都一样，但对第二个参数：apply传入的是一个参数数组，也就是将多个参数组合成为一个数组传入，而call方法的参数传递方法如下 func.call(func1,var1,var2,var3)

对应的apply写法为：func.apply(func1,[var1,var2,var3])；

同时使用apply的好处是可以直接将当前函数的arguments对象作为apply的第二个参数传入。

exp:

function add(c,d){

        console.log(this.a + this.b + c + d);

    }

    var o ={a:1,b:3};

    add.call(o,5,7);    //1+3+5+7=16

    //传参的时候是扁平的把每个参数传进去

    add.apply(o,[10,20]);   //1+3+10+20=34

//传参的时候是把参数作为一个数组传进去。

    //什么时候使用call或者apply

    function bar(){

        console.log(Object.prototype.toString.call(this));

        // 用来调用一些无法直接调用的方法

}

bar.call(7); //"[object Number]"

**5. 数组和对象有哪些原生方法，列举一下，分别是什么含义，比如连接两个数组用哪个方法，删除数组的指定项和重新组装数组(操作数据的重点)。**

1)数组：

length设置或返回 数组中元素的数目。

push() :向数组的末尾添加一个或多个元素，并返回新的长度，也就是添加元素后的数组长度。

shift():用于把数组的第一个元素从其中删除，并返回第一个元素的值。

unshift()：向数组的开头添加一个或更多元素，并返回新的长度。

pop():用于删除并返回数组的最后一个元素。

splice()：用于插入、删除或替换数组的元素。

concat():方法用于连接两个或多个数组。

join():用于把数组中的所有元素放入一个字符串。元素是通过指定的分隔符进行分隔的。

toString():方法可把数组转换为字符串，并返回结果。

reverse():方法用于颠倒数组中元素的顺序。

slice():方法可从已有的数组中返回选定的元素。

sort():方法用于对数组的元素进行排序（从小到大）。

indexOf():返回获取项在数组中的索引

lastIndexOf():返回获取项在数组中出现的最后一次索引

forEach():循环遍历数组 参数是一个匿名函数 默认返回为undefined

map()：循环遍历数组 参数是一个匿名函数

对象：

charAt(): 返回在指定位置的字符。

charCodeAt(): 返回在指定的位置的字符的 Unicode 编码。

concat():连接字符串。

indexOf():检索字符串。

match(): 找到一个或多个正则表达式的匹配。

replace():替换与正则表达式匹配的子串。

search(): 检索与正则表达式相匹配的值。

slice(): 提取字符串的片断，并在新的字符串中返回被提取的部分。

split(): 把字符串分割为字符串数组。

toLocaleLowerCase(): 把字符串转换为小写。

toLocaleUpperCase(): 把字符串转换为大写。

toLowerCase(): 把字符串转换为小写。

toUpperCase(): 把字符串转换为大写。

substr(): 从起始索引号提取字符串中指定数目的字符。

substring(): 提取字符串中两个指定的索引号之间的字符。

2)连接两个数组用concat（)方法，删除数组中的指定项使用shift（）（删除数组中的第一个元素并返回第一个元素），或pop（）（删除数组中最后个元素并返回该元素），splice()(删除从任意位置开始的元素，然后返回被删除的项目，arrayObject.splice(index,howmany,item1,.....,itemX)，他会改变原数组)。

**3）**

**6. 怎样避免全局变量污染?ES5严格模式的作用，ES6箭头函数和ES5普通函数一样吗?**

var my={};

my.name={

big\_name:"zhangsan",

small\_name:"lisi"

};

my.work={

school\_work:"study",

family\_work:"we are"

};

利用匿名函数将脚本包裹起来

(function(){

var exp={};

var name="aa";

exp.method=function(){

return name;

};

window.ex=exp;

})();

2） 严格模式的作用：

1、消除javascript语法的一些不合理、不严谨之处，减少一些怪异行为；

2、消除代码运行的一些不安全性，促进代码运行的安全；

3、提高编译器效率，增加运行速度；

4、为未来新版本的javascript做好铺垫

IE10\chrome\火狐等主流浏览器已经全面支持

3）箭头函数是ES6提供的一种语法，简化了普通函数写法，并且将当前上下文对象传递给创建的函数。此外箭头函数全都是匿名函数，不能用于构造函数，不具有arguments对象，不能拥有generator函数，不具备prototype原型对象，不具有super对象(指向当前对象的原型对象)，不具有new.target（new.target属性允许你检测函数或构造方法是否是通过new运算符被调用的。在通过new运算符被初始化的函数或构造方法中，new.target返回一个指向构造方法或函数的引用。在普通的函数调用中，new.target 的值是undefined。）

**面向对象**

**1. JS 模块包装格式都用过哪些，CommonJS、AMD、CMD。定义一个JS 模块代码，最精简的格式是怎样。**

**2. JS 怎么实现一个类。怎么实例化这个类。**

**3. 理解闭包吗?请讲一讲闭包在实际开发中的作用;闭包建议频繁使用吗?**

**4. 说一下了解的js 设计模式，解释一下单例、工厂、观察者。**

**5. ajax 跨域有哪些方法，jsonp 的原理是什么，如果页面编码和被请求的资源编码不一致如何处理?**

**「 开源工具 」**

**1)是否了解开源的架构工具 bower、npm、yeoman、gulp、webpack，有无用过，有无写过，一个 npm 的包里的 package.json 具备的必要的字段都有哪些(名称、版本号，依赖)**

**2)github常用不常用，关注过哪些项目**

**3)会不会用 ps 扣图，png、jpg、gif 这些图片格式解释一下，分别什么时候用。如何优化图像、图像格式的区别**

**4)说一下你常用的命令行工具**

**5)会不会用git，说上来几个命令，说一下git和svn的区别，有没有用git解决过冲突**

**「 计算机网络基础 」**

**1)说一下HTTP 协议头字段说上来几个，是否尽可能详细的掌握HTTP协议。一次完整的HTTP事务是怎样的一个过程?**

**2)cookies 是干嘛的，服务器和浏览器之间的 cookies 是怎么传的，httponly 的 cookies 和可读写的 cookie 有什么区别，有无长度限制**

**请描述一下cookies，sessionStorage和localStorage的区别**

**3)从敲入 URL 到渲染完成的整个过程，包括 DOM 构建的过程，说的约详细越好。**

**4)是否了解Web注入攻击，说下原理，最常见的两种攻击(XSS 和 CSRF)了解到什么程度。**

**5)是否了解公钥加密和私钥加密。如何确保表单提交里的密码字段不被泄露。验证码是干嘛的，是为了解决什么安全问题。**

**6)编码常识：文件编码、URL 编码、Unicode编码 什么含义。一个gbk编码的页面如何正确引**

**「 前端框架 」**

**1) 对 MVC、MVVM的理解**

**2) vue、angularjs等 相对于 jQuery在开发上有什么优点?**

**3)前后分离的思想了解吗?**

**4)你上一个项目都用到了那些方法优化js的性能?**

**5)angular的生命周期?**

**6)说一下你对vue和vuex的使用方法，vue的组件复用机制**

**考察学习能力和方法**

**1)你每天必须登录的网站(前端技术相关)是什么?**

**2)前端技术方面看过哪些书，有无笔记，都有哪些收获。**

**3)收藏了哪些代码片段?有想过开源自己的代码嘛?**

**4)怎么理解前端技术的大趋势?自己再做哪方面的知识储备?**

**5)是否了解或精通其他(后端)的编程语言?**

**6)做项目有没有遇到哪些印象深刻的技术攻关，具体遇到什么问题，怎么找答案的，最后怎么解的。**

**7)对以后自己的前端职业路线，怎么规划？**

开放性问题(重要)

这些问题往往决定你是否最终被录用或者等到终轮面试，技术点回答错了不要紧，人脑不是机器，是可以恶补的。

但如果你没有思想和独到的思路，基础挖的再深，可能也打动不了面试官，因为比你基础好的一大堆，但每个人的个性思想却是不同的。

1. 如果需要你加班，你会加吗，抵触吗?

其实你肯定抵触，但你肯定要回答如果项目需要肯定会加。

2. 一个小项目让你自己负责搭建底层一些架构，你能胜任吗?

回答例如：我肯定愿意尝试，并做到最优的选择方案出来。

3. 如果项目拖太久，你情绪低落或者厌烦了怎么调节?

回答就是， 你结合自身挑着好听的说就行，就像聊天。

4. 你建议自己造轮子，还是利用开源的轮子?

回答：根据实际情况而定，如果开源完全满足 可以自己二次开发就好，大大缩短开发周期，

如果实在没有契合度很高的，可以花费几个工作日尝试造轮。