一个优秀的前端开发人员需要具备哪些特质？

学习能力、代码可维护性、代码规范性、解决问题的能力、沟通能力

常见的布局方式有哪些？

table布局、流式布局、弹性布局、响应式布局

**css**预加载的使用？

他们是动态的样式语言，是CSS预处理器,CSS上的一种抽象层。他们是一种特殊的语法/语言而编译成CSS。

变量符不一样，less是@，而Sass是$;

Sass支持条件语句，可以使用if{}else{},for{}循环等等。而Less不支持;

Sass是基于Ruby的，是在服务端处理的，而Less是需要引入less.js来处理Less代码输出Css到浏览器

<https://juejin.im/post/5a2bc28f6fb9a044fe464b19>

懒加载和预加载： **<https://www.jianshu.com/p/4876a4fe7731>**

常见的解决浏览器兼容性问题的方式有哪些？

**<https://juejin.im/post/59a3f2fe6fb9a0249471cbb4>**

**<https://github.com/Wscats/Good-text-Share/issues/61>**

移动端屏幕适配

**<https://www.w3cplus.com/mobile/lib-flexible-for-html5-layout.html>**

介绍一下标准的**css**的盒子模型？

**<https://blog.csdn.net/exialym/article/details/51637184>**

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？

1、设置margin:0 auto;width:。。。

2、确定容器的宽高，这里宽度是必须的，高度可以不设，设置外层的上外边距和左外边距分别是宽高的一半。

实现居中关键在于 margin设置与 position:relative.

.div {

width:500px ;

height:300px;

margin: -150px 0 0 -250px;

position:relative;

left:50%;

top:50%;

}

c.position:absolute;

top: 50%;

left: 50%; 只能把div定位在以红色圈为起点的位置，加上margin:-100px 0px 0px -100

3、position+transform

**display**设置为**inline-block**时，**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（回车空格等等）的影响，这些空白也会被应用样式，占据空间，所以会有间隔

解决：设置ul的font-size为0，缺陷是必须重新在li中去设置字体大小

请解释下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？

a.添加新的元素 、应用 clear：both;

b.父级定义 overflow: auto（注意：是父级div也就是这里的 div.outer） 一个对seo比较友好，另个hidden对seo不是太友好

在IE6中还需要触发 hasLayout ，例如 zoom：1；

c.据说是最高大上的方法 :after

方法：（注意：作用于浮动元素的父亲）IE6-7不支持:after，

使用 zoom:1触发 hasLayout

{zoom:1;} /\*==for IE6/7 Maxthon2==\*/

:after {clear:both;content:'.';display:block;width: 0;height: 0;visibility:hidden;}

d.使用 br标签和其自身的 html属性,<br clear="all" /> clear=“all | left | right | none” 属性

e.父元素也设置浮动

f.父元素设置display:table 盒模型属性已经改变，由此造成的一系列问题，得不偿失，不推荐使用

**margin**和**padding**分别适合什么场景使用？

何时应当使用margin：

（1）需要在border外侧添加空白时，

（2）空白处不需要有背景（色）时，

（3）上下相连的两个盒子之间的空白需要相互抵消时，比如15px+20px的margin，将得到20px的空白（注意地方见第三点）。

何时应当使用padding

（1）需要在border内侧添加空白时（往往是文字与边框距离的设置），

（2）空白处需要背景（色）时，

（3）上下相连的两个盒子之间的空白希望等于两者之和时，比如15px+20px的padding，将得到35px的空白。

margin使用时应该注意的地方

margin在垂直方向上相邻的值相同时会发生叠加，水平方向上相邻的值会相加。margin取负值时，在垂直方向上，两个元素的边界仍然会重叠。但是，此时一个为正值，一个为负值，并不是取其中较大的值，而是用正边界减去负边界的绝对值，也就是说，把正的边界值和负的边界值相加。

什么是响应式设计？响应式设计的基本原理是什么？如何兼容较低版本的**IE**？

答：一个网站能够兼容多个终端——而不是为每个终端做一个特定的版本。

优点：

面对不同分辨率设备灵活性强

能够快捷解决多设备显示适应问题

缺点：

兼容各种设备工作量大，效率低下

代码累赘，会出现隐藏无用的元素，加载时间加长

其实这是一种折中性质的设计解决方案，多方面因素影响而达不到最佳效果

一定程度上改变了网站原有的布局结构，会出现用户混淆的情况

有一个高度自适应的**div**。里面有**2**个**div**，一个高度**100px**，希望另一个填满剩下的高度？

外层box-sizing: border-box; 同时设置padding: 100px 0 0；

内层100像素高的元素向上移动100像素，或使用absolute定位防止占据空间；

另一个元素直接height: 100%;

外层position: relative；

百分百自适应元素直接position: absolute; top: 100px; bottom: 0; left: 0

**let**、**const**、**var**的区别？

**<http://www.cnblogs.com/ghostwu/p/7286053.html>**

**nll**和**undefined**的区别？

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

（1） 作为函数的参数，表示该函数的参数不是对象。

（2） 作为对象原型链的终点。

undefined表示"缺少值"，就是此处应该有一个值，但是还没有定义

（1）变量被声明了，但没有赋值时，就等于undefined。

（2) 调用函数时，应该提供的参数没有提供，该参数等于undefined。

（3）对象没有赋值的属性，该属性的值为undefined。

（4）函数没有返回值时，默认返回undefined。

什么是闭包（**closure**）？为什么要用它？使用举例

闭包就是能够读取其他函数内部变量的函数

由于在Javascript语言中，只有函数内部的子函数才能读取局部变量，因此可以把闭包简单理解成“定义在一个函数内部的函数”。

所以，在本质上，闭包就是将函数内部和函数外部连接起来的一座桥梁。

闭包的用途：

闭包可以用在许多地方。它的最大用处有两个，一个是前面提到的可以读取函数内部的变量，另一个就是让这些变量的值始终保持在内存中

重用变量又不能造成全局污染

举例：朋友圈点赞、计数器

**js**代码中**“use strict”**是什么意思？使用它区别是什么？

进入”严格模式”的标志，老版本的浏览器会把它当作一行普通字符串，加以忽略

将”use strict”放在脚本文件的第一行，则整个脚本都将以”严格模式”运行。如果这行语句不在第一行，则无效，整个脚本以”正常模式”运行。如果不同模式的代码文件合并成一个文件，这一点需要特别注意。

(严格地说，只要前面不是产生实际运行结果的语句，”use strict”可以不在第一行，比如直接跟在一个空的分号后面。)

将”use strict”放在函数体的第一行，则整个函数以”严格模式”运行

因为第一种调用方法不利于文件合并，所以更好的做法是，借用第二种方法，将整个脚本文件放在一个立即执行的匿名函数之中

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

– 消除代码运行的一些不安全之处，保证代码运行的安全；

– 提高编译器效率，增加运行速度；

– 为未来新版本的Javascript做好铺垫。

**js**中有一个函数，执行对象查找时，永远不会去查找原型，这个函数是什么？

hasOwnProperty

**js**延迟加在的方式有哪些？

1. <script type="text/javascript" src="id=my"></script>

<script type=”text/javascript”>

setTimeout(“document.getElementById(‘my').src='include/php100.php'; “,3000);//延时3秒

</script>

2. <span id=”L4EVER”>LOADING…</span>

<span id=”AD\_L4EVER”>你的JS代码在这里!</span >

<script>L4EVER.innerHTML=AD\_L4EVER.innerHTML;AD\_L4EVER.innerHTML=”";</script>

4. js外部引入的文件放到页面底部，来让js最后引入，从而加快页面加载速度

同步和异步的区别？

**<https://www.jianshu.com/p/561db8ff3e7a>**

**document.write**和**innerHTNML**的区别？

1.document.write是直接写入到页面的内容流，如果在写之前没有调用document.open, 浏览器会自动调用open。每次写完关闭之后重新调用该函数，会导致页面被重写。

2.innerHTML则是DOM页面元素的一个属性，代表该元素的html内容。你可以精确到某一个具体的元素来进行更改。如果想修改document的内容，则需要修改document.documentElement.innerElement。

3.两者都可动态包含外部资源如JavaScript文件

通过document.write插入<script></script>元素会自动执行其中的脚本；

大多数浏览器中，通过innerHTML插入<script></script>元素并不会执行其中的脚本

innerHTML很多情况下都优于document.write，其原因在于其允许更精确的控制要刷新页面的那一个部分。

对**promise**的理解

<https://zhuanlan.zhihu.com/p/23907711>

引起内存泄漏的原因：

**<https://juejin.im/post/5a8e7f6df265da4e832677ec>**

谈谈你对**this**的理解 **this**的使用场景、**this**绑定、**this**丢失

答案：参考晚自习串讲**ppt**

需求：实现一个页面操作不会整页刷新的网站，并且能在浏览器的前进，后退时正确响应。给出你的技术实现方案？

用cookie或者localStorage来记录应用的状态即可，刷新页面时读取一下这个状态，然后发送相应ajax请求来改变页面即可

HTML5里引用了新的API，就是history.pushState和history.replaceState，就是通过这个接口做到无刷新改变页面URL的

虽然ajax可以无刷新改变页面内容，但无法改变页面URL

其次为了更好的可访问性，内容发生改变后，改变URL的hash。但是hash的方式不能很好的处理浏览器的前进、后退等问题

有的浏览器引入了onhashchange的接口，不支持的浏览器只能定时去判断hash是否改变

再有，ajax的使用对搜索引擎很不友好，往往蜘蛛爬到的区域是空的

为了解决传统ajax带来的问题，HTML5里引入了新的API，即：history.pushState, history.replaceState

可以通过pushState和replaceState接口操作浏览器历史，并且改变当前页面的URL。

pushState是将指定的URL添加到浏览器历史里，replaceState是将指定的URL替换当前的URL。

如何调用？

var state = { title: title, url: options.url, otherkey: othervalue};window.history.pushState(state, document.title, url);

state对象除了要title和url之外，也可以添加其他的数据，比如：还想将一些发送ajax的配置给保存起来。

如何响应浏览器的前进、后退操作

window对象上提供了onpopstate事件，上面传递的state对象会成为event的子对象，这样就可以拿到存储的title和URL了。

window.addEventListener('popstate', function(e){ if (history.state){ var state = e.state; //do something(state.url, state.title); }}, false);

这样就可以结合ajax和pushState完美的进行无刷新浏览了。

js的数据类型都有哪些？

原始类型、引用类型；区别？

什么是**Ajax**，优缺点？使用场合？

Ajax是异步JavaScript和XML，用于在Web页面中实现异步数据交互。

　　优点：可以使得页面不重载全部内容的情况下加载局部内容，降低数据传输量

避免用户不断刷新或者跳转页面，提高用户体验

缺点：对搜索引擎不友好（

要实现ajax下的前后退功能成本较大

可能造成请求数的增加

跨域问题限制

如何解析**JSON**？**(**面试必问**)**

**<https://www.jianshu.com/p/43517c69fe48>**

**js**的**typeof**返回哪些数据类型？

**split() join()**的区别？

join() 方法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，指定分隔符方法join(“#");其中#可以是任意与之相反的是split()方法：用于把一个字符串分割成字符串数组.

数组方法**pop() push() unshift() shift()?**

push和pop方法，这两个方法只会对数组从尾部进行压入或弹出，而且是在原数组进行操作，任何的改动都是会影响到操作的数组。push(args)可以每次压入多个元素，并返回更新后的数组长度。pop()函数每次只会弹出最后一个结尾的元素，并返回弹出的元素，如果是对空组数调用pop()则返回undefined。 如果参数是数组则是将整个数组当做一个元素压入到原来的数组当中。并不会产生类似concat合并数组时产生的“拆分现象”

unshift和shift这两个方法都是通过对数组的头部进行的操作，其他基本跟push和pop类似

shift:从集合中把第一个元素删除，并返回这个元素的值。

unshift: 在集合开头添加一个或更多元素，并返回新的长度

push:在集合中添加元素，并返回新的长度

pop:从集合中把最后一个元素删除，并返回这个元素的值

**ES6**的了解？

参考：**http://es6.ruanyifeng.com/**

**js**的本地对象，内置对象和宿主对象？

本地对象：Object、Function、Array、String、Boolean、Number、Date、RegExp、Error、EvalError、RangeError、ReferenceError、SyntaxError、TypeError、URIError官方定义好了的对象

内置对象： Global 和 Math，内置对象是本地对象的一种

宿主对象：所有的BOM和DOM对象都是宿主对象，是那些官方未定义，你自己构建的对象加上DOM和BOM对象组成的

列举所了解的前端框架并简述？

<https://cn.vuejs.org/v2/guide/comparison.html>

你如何对网站的文件和资源进行优化？期待的解决方法包括？

A、文件合并，减少http请求，合并JavaScript和CSS文件、CSS Sprites、图像映射 （Image Map）和使用Data URI来编码图片

B、文件最小化/文件压缩，减少文件下载的体积；常用的工具是YUI Compressor

C、使用 CDN 托管，尽可能避开互联网上有可能影响数据传输速度和稳定性的瓶颈和环节，使内容传输的更快、更稳定

D、缓存的使用（多个域名来提供缓存）

E、GZIP 压缩你的 JS 和 CSS 文件

**js**线程与进程的区别？

一个程序至少有一个进程,一个进程至少有一个线程.

线程的划分尺度小于进程，使得多线程程序的并发性高。

另外，进程在执行过程中拥有独立的内存单元，而多个线程共享内存，从而极大地提高了程序的运行效率。

线程在执行过程中与进程还是有区别的。每个独立的线程有一个程序运行的入口、顺序执行序列和程序的出口。但是线程不能够独立执行，必须依存在应用程序中，由应用程序提供多个线程执行控制。

从逻辑角度来看，多线程的意义在于一个应用程序中，有多个执行部分可以同时执行。但操作系统并没有将多个线程看做多个独立的应用，来实现进程的调度和管理以及资源分配。这就是进程和线程的重要区别。

请解释一下什么是**“**语义化的 **HTML”**？

语义化的好处：1：去掉或样式丢失的时候能让页面呈现清晰的结构：2.屏幕阅读器（如果访客有视障）会完全根据你的标记来“读”你的网页.3.PDA、手机等设备可能无法像普通电脑的浏览器一样来渲染网页（通常是因为这些设备对CSS的支持较弱4.搜索引擎的爬虫也依赖于标记来确定上下文和各个关键字的权重.5.你的页面是否对爬虫容易理解非常重要,因为爬虫很大程度上会忽略用于表现的标记, 而只注重语义标记.6.便于团队开发和维护语义化的HTML就是：标题用h1-h6，文字段落用p，列表用ul li，大致如此

为什么利用多个域名来提供网站资源会更有效？

浏览器同一时间可以从一个域名下载多少资源？你的浏览器能同时保持对一个域名的多少连接？

三个最主流的原因:

1. CDN缓存更方便

2. 突破浏览器并发限制 (你随便挑一个 G家的 url: https://lh4.googleusercontent.com/- si4dh2myPWk/T81YkSi\_\_AI/AAAAAAAAQ5o/LlwbBRpp58Q/w497-h373/IMG\_20120603\_163233.jpg, 把前面的 lh4换成 lh3,lh6 啥的，都照样能够访问，像地图之类的需要大量并发下载图片的站点，这个非常重要。)

3. Cookieless, 节省带宽，尤其是上行带宽 一般比下行要慢。。。

还有另外两个非常规原因:

4. 对于UGC的内容和主站隔离，防止不必要的安全问题( 上传js窃取主站cookie之类的) 。

正是这个原因要求用户内容的域名必须不是自己主站的子域名，而是一个完全独立的第三方域名。

5. 数据做了划分，甚至切到了不同的物理集群，通过子域名来分流比较省事. ^\_^ 这个可能被用的不多。

PS: 关于Cookie的问题，带宽是次要的，安全隔离才是主要的。

关于多域名，也不是越多越好，虽然服务器端可以做泛解释，浏览器做dns解释也是耗时间的，而且太多域名，如果要走 https的话，还有要多买证书和部署的问题，^\_^。

如果你参与到一个项目中，发现他们使用 **Tab** 来缩进代码，但是你喜欢空格，你会怎么做

1.建议这个项目使用像 EditorConfig (http://editorconfig.org/) 之类的规范

2.为了保持一致性，接受项目原有的风格

3.直接使用 VIM 的 retab 命令

**data-**属性的作用是什么？

data-为前端开发者提供自定义的属性，这些属性集可以通过对象的dataset属性获取，不支持该属性的浏览器可以通过 getAttribute方法获取

<div data-author="david" data-time="2011-06-20" data-comment-num="10">...</div>

div.dataset.commentNum; // 10

需要注意的是，data-之后的以连字符分割的多个单词组成的属性，获取的时候使用驼峰风格。并不是所有的浏览器都支持.dataset属性，测试的浏览器中只有Chrome和Opera支持。

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

sessionStorage 和 localStorage 是HTML5 Web Storage API 提供的，可以方便的在web请求之间保存数据。有了本地数据，就可以避免数据在浏览器和服务器间不必要地来回传递。

sessionStorage、localStorage、cookie都是在浏览器端存储的数据，其中sessionStorage的概念很特别，引入了一个“浏览器窗口”的概念。sessionStorage是在同源的同窗口（或tab）中，始终存在的数据。也就是说只要这个浏览器窗口没有关闭，即使刷新页面或进入同源另一页面，数据仍然存在。关闭窗口后，sessionStorage即被销毁。同时“独立”打开的不同窗口，即使是同一页面，sessionStorage对象也是不同的

cookies会发送到服务器端。其余两个不会。

Microsoft指出InternetExplorer8增加cookie限制为每个域名50个，但IE7似乎也允许每个域名50个cookie。

　　Firefox每个域名cookie限制为50个。

　　Opera每个域名cookie限制为30个。

Firefox和Safari允许cookie多达4097个字节，包括名（name）、值（value）和等号。

　　Opera允许cookie多达4096个字节，包括：名（name）、值（value）和等号。

　　InternetExplorer允许cookie多达4095个字节，包括：名（name）、值（value）和等号。

描述下 **“reset” CSS** 文件的作用和使用它的好处。

因为浏览器的品种很多，每个浏览器的默认样式也是不同的，所以定义一个css reset可以使各浏览器的默认样式统一。

如何视觉隐藏网页内容，只让它们在屏幕阅读器中可用？

1.display:none;的缺陷

搜索引擎可能认为被隐藏的文字属于垃圾信息而被忽略

屏幕阅读器（是为视觉上有障碍的人设计的读取屏幕内容的程序）会忽略被隐藏的文字。

2. visibility: hidden ;的缺陷

这个大家应该比较熟悉就是隐藏的内容会占据他所应该占据物理空间

3.overflow:hidden;一个比较合理的方法

.texthidden { display:block;/\*统一转化为块级元素\*/ overflow: hidden; width: 0; height: 0; }

就像上面的一段CSS所展示的方法，将宽度和高度设定为0，然后超过部分隐藏，就会弥补上述一、二方法中的缺陷，也达到了隐藏内容的目的。

解释下事件代理？

JavaScript事件代理则是一种简单的技巧，通过它你可以把事件处理器添加到一个父级元素上，这样就避免了把事件处理器添加到多个子级元素上。

当我们需要对很多元素添加事件的时候，可以通过将事件添加到它们的父节点而将事件委托给父节点来触发处理函数。这主要得益于浏览器的事件冒泡机制。

事件代理用到了两个在JavaSciprt事件中常被忽略的特性：事件冒泡以及目标元素。

function getEventTarget(e) {

e = e || window.event;

return e.target || e.srcElement;

}

解释下原型继承的原理？

function getProperty(obj, prop) {

if (obj.hasOwnProperty(prop)) {

return obj[prop];

}else if (obj.\_\_proto\_\_ !== null) {

return getProperty(obj.\_\_proto\_\_, prop);

}else{

return undefined;

}

}

实操部分

用原型链的方式给**Array**对象添加一个数组去重的方法？

定义一个方法，对所有传入的数字参数的第三位小数进行四舍五入，使得返回值保留两位小数，不够的补**0**

定义一个方法，实现阶乘

定义一段代码，页面载入完成**1**分钟后非缓存模式刷新当前页面

把**Object**的实例化对象**A**、**B**、**C**合并赋值给对象**C**

写一个**function** 清楚字符串前后的空格（兼容所有浏览器）

获取指定范围内**N**个不重复的随机数

对事件的理解，如何阻止事件默认行为和事件冒泡？

默认行为：event.preventDefault();

冒泡：event.stopPropregation();

event.cancelBubble();

进阶部分

**$interval**和**setInterval**区别？

用过**ng-service**和**ng-factory**吗？请简单描述。

Factory 就是创建一个对象，为它添加属性，然后把这个对象返回出来。你把 service 传进 controller 之后，在 controller 里这个对象里的属性就可以通过 factory 使用了。

Service 是用"new"关键字实例化的。因此，你应该给"this"添加属性，然后 service 返回"this"。你把 service 传进 controller 之后，在controller里 "this" 上的属性就可以通过 service 来使用了。

**Web**前端优化面试要点（面试必问）：

**<https://mp.weixin.qq.com/s/9A0egnG1ripm08wGRkpfug>**

（1）. CSS Sprites 合成图片

优点：减少HTTP请求次数，图片合成后总大小变小，加快加载

缺点：不利于后期修改，每次修改单个图片都要重新合成一次；不利于代码语义化

折中：按照使用频率、功能类型的不同合成几个级别不同的图片

（2）. 合并css, js文件

优点：减少HTTP请求次数，加快加载

缺点：牺牲了代码组织结构，使单个文件内部混乱

折中：适当分开，尽量合并

（3）. 压缩文件大小

代码文件：用压缩工具压缩，去掉空白、无用、重复代码； 图片文件：降低分辨率、改变格式（PNG优先）、压缩图片质量等，也可以使用专业的压缩工具

优点：缩小体积，加快加载

缺点：没有明显缺点，如果一定要说的话，压缩后的代码可读性差（不过一般都会留一份压缩前的源码）

（4）. 结构（HTML）样式（CSS）行为（JS）分离

优点：

使项目结构更清晰，易于维护，易于对各部分进行单独的处理而不影响其他部分。

提高代码可复用性

浏览器会单独缓存css, js文件，加快加载

缺点：有缺点吗？对了，demo之类的小项目可以不用分离了，多此一举

（5）. CSS放在页面开头（<head>内部），JS放在页面末尾（</body>之前）

DOM树的生成是受JS代码执行的影响的，JS代码会“阻塞”页面的渲染，详细请读《JavaScript权威指南》(P320-P322)

（6）. 使用CDN（内容发布网络）

优点：CDN会帮你找到距离用户电脑最近的服务器上的资源文件，更够更快速地传到相应的浏览器上。

缺点：

不利于离线开发，也无法对远程的资源进行控制（比如添加一些自定义的修改）

有些CDN服务在某些国家和地区无法访问，比如在天朝访问使用了google cdn的网站时就会出现很多问题

（7）. 尽量减少浏览器reflow

这里设计关于Reflow和Repaint的知识。Reflow简单来讲就是浏览器重新渲染页面布局，这可能是由于一个很小的节点的改变引发的，作用很小，代价很大，所以应尽量避免。

措施：

减小不必要的DOM树深度，也就是说，不必要的话，你的DOM树不要太多层了，其中一层的布局变化将引起整棵树的重新渲染。

尽量使用对class进行操作来动态改变元素样式，而不是直接对元素css进行更改。

需要使用动画的元素应将其position设置为absolute或者fixed，使其脱离文档流。

避免过于复杂的css选择器，尤其要避免使用过多后代选择器，太多层CPU开销很大。

不要用表格进行页面布局，表格式牵一发动全身的布局方式。

不要用css表达式

（8）. 使用<link>，避免使用@import加载css

优点：@import的css文件无法同步并行加载，必须等一个加载完才能加载下一个，势必造成速度问题，而link引用的多个stylesheet则可以同步并行加载，速度更快

缺点：暂时没想到

（9）. 尽量少使用css filter达到滤镜效果，可以用png图片

（10）. 能用HTML做的事，不要让css、js来做，能用css做的事，不要让js来做

（11）. 使用iconfont代替图片图标

如何解决跨域问题？原理？

JSONP：

原理是：动态插入script标签，通过script标签引入一个js文件，这个js文件载入成功后会执行我们在url参数中指定的函数，并且会把我们需要的json数据作为参数传入。

由于同源策略的限制，XmlHttpRequest只允许请求当前源（域名、协议、端口）的资源，为了实现跨域请求，可以通过script标签实现跨域请求，然后在服务端输出JSON数据并执行回调函数，从而解决了跨域的数据请求。

优点是兼容性好，简单易用，支持浏览器与服务器双向通信。缺点是只支持GET请求。

JSONP：json+padding（内填充），顾名思义，就是把JSON填充到一个盒子里

<script>

function createJs(sUrl){

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

oScript.type = 'text/javascript';

oScript.src = sUrl;

document.getElementsByTagName('head')[0].appendChild(oScript);

}

createJs('jsonp.js');

box({

'name': 'test'

});

function box(json){

alert(json.name);

}

</script>

CORS

服务器端对于CORS的支持，主要就是通过设置Access-Control-Allow-Origin来进行的。如果浏览器检测到相应的设置，就可以允许Ajax进行跨域的访问。

通过修改document.domain来跨子域

将子域和主域的document.domain设为同一个主域.前提条件：这两个域名必须属于同一个基础域名!而且所用的协议，端口都要一致，否则无法利用document.domain进行跨域

主域相同的使用document.domain

使用window.name来进行跨域

window对象有个name属性，该属性有个特征：即在一个窗口(window)的生命周期内,窗口载入的所有的页面都是共享一个window.name的，每个页面对window.name都有读写的权限，window.name是持久存在一个窗口载入过的所有页面中的

使用HTML5中新引进的window.postMessage方法来跨域传送数据

还有flash、在服务器上设置代理页面等跨域方式。个人认为window.name的方法既不复杂，也能兼容到几乎所有浏览器，这真是极好的一种跨域方法。

**XML**和**JSON**的区别？

(1).数据体积方面。

JSON相对于XML来讲，数据的体积小，传递的速度更快些。

(2).数据交互方面。

JSON与JavaScript的交互更加方便，更容易解析处理，更好的数据交互。

(3).数据描述方面。

JSON对数据的描述性比XML较差。

(4).传输速度方面。

JSON的速度要远远快于XML。

TCP协议

TCP（Transmission Control Protocol，传输控制协议）是基于连接的协议，也就是说，在正式收发数据前，必须和对方建立可靠的连接。一个TCP连接必须要经过三次“对话”才能建立起来，其中的过程非常复杂。

TCP充分实现了数据传输时各种控制功能，可以进行丢包的重发控制，还可以对次序乱掉的分包进行顺序控制。TCP通过检验和、序列号、确认应答、重发控制、连接管理以及窗口控制等机制实现可靠性传输。

UDP协议

UDP（User Data Protocol，用户数据报协议）是与TCP相对应的协议。它是面向非连接的协议，它不与对方建立连接，而是直接就把数据包发送过去！ UDP适用于一次只传送少量数据、对可靠性要求不高的应用环境。

TCP协议和UDP协议的区别

(1).TCP面向连接（如打电话要先拨号建立连接）；UDP是无连接的，即发送数据之前不需要建立连接；

(2).TCP提供可靠的服务。也就是说，通过TCP连接传送的数据，无差错，不丢失，不重复，且按序到达；UDP尽最大努力交付，即不保证可靠交付；

(3).TCP面向字节流，实际上是TCP把数据看成一连串无结构的字节流；UDP是面向报文的

UDP没有拥塞控制，因此网络出现拥塞不会使源主机的发送速率降低（对实时应用很有用，如IP电话，实时视频会议等）；

(4).每一条TCP连接只能是点到点的;UDP支持一对一，一对多，多对一和多对多的交互通信；

(5).TCP首部开销20字节；UDP的首部开销小，只有8个字节；

(6).TCP的逻辑通信信道是全双工的可靠信道；UDP则是不可靠信道。

**Ajax**是什么？**Ajax**常用的技术列举几个？**Ajax**的缺点有什么？

AJAX定义

ajax的全称是AsynchronousJavascript+XML，是指一种创建交互式网页应用的网页开发技术。即异步传输+js+xml。通过在后台与服务器进行少量数据交换，AJAX 可以使网页实现异步更新。这意味着可以在不重新加载整个网页的情况下，对网页的某部分进行更新。而传统的网页（不使用 AJAX）如果需要更新内容，必须重载整个网页页面。

AJAX常用的技术

基于 web 标准（standards-based presentation）XHTML+CSS的表示；

使用 DOM（Document Object Model）进行动态显示及交互；

使用 XML 和 XSLT 进行数据交换及相关操作；

使用 XMLHttpRequest 进行异步数据查询、检索；

使用 JavaScript 将所有的东西绑定在一起

类似于DHTML或LAMP，AJAX不是指一种单一的技术，而是有机地利用了一系列相关的技术。

AJAX的优缺点

1.优点

无刷新更新数据：AJAX最大优点就是能在不刷新整个页面的前提下与服务器通信维护数据。这使得Web应用程序更为迅捷地响应用户交互，并避免了在网络上发送那些没有改变的信息，减少用户等待时间，带来非常好的用户体验。

异步与服务器通信：AJAX使用异步方式与服务器通信，不需要打断用户的操作，具有更加迅速的响应能力。优化了Browser和Server之间的沟通，减少不必要的数据传输、时间及降低网络上数据流量。

前端和后端负载平衡：AJAX可以把以前一些服务器负担的工作转嫁到客户端，利用客户端闲置的能力来处理，减轻服务器和带宽的负担，节约空间和宽带租用成本。并且减轻服务器的负担，AJAX的原则是“按需取数据”，可以最大程度的减少冗余请求和响应对服务器造成的负担，提升站点性能。

基于标准被广泛支持：AJAX基于标准化的并被广泛支持的技术，不需要下载浏览器插件或者小程序，但需要客户允许JavaScript在浏览器上执行。

界面与应用分离：Ajax使WEB中的界面与应用分离（也可以说是数据与呈现分离），有利于分工合作、减少非技术人员对页面的修改造成的WEB应用程序错误、提高效率、也更加适用于现在的发布系统。

2.缺点

对浏览器机制的破坏：AJAX干掉了Back和History功能，在动态更新页面的情况下，用户无法回到前一个页面状态，因为浏览器仅能记忆历史记录中的静态页面。一个被完整读入的页面与一个已经被动态修改过的页面之间的差别非常微妙；用户通常会希望单击后退按钮能够取消他们的前一次操作，但是在Ajax应用程序中，这将无法实现。

AJAX的安全问题：Ajax技术就如同对企业数据建立了一个直接通道，这使得开发者在不经意间会暴露比以前更多的数据和服务器逻辑。

对搜索引擎支持较弱：如果使用不当，AJAX会增大网络数据的流量，从而降低整个系统的性能。

破坏程序的异常处理机制

违背URL和资源定位的初衷：如果采用了Ajax技术，也许不同的人在同一个URL地址下看到的内容是不同的。这个和资源定位的初衷是相背离的。

AJAX不能很好支持移动设备。

10、对前端模块化的认识

AMD 是 RequireJS 在推广过程中对模块定义的规范化产出。

CMD 是 SeaJS 在推广过程中对模块定义的规范化产出。

AMD 是提前执行，CMD 是延迟执行。

AMD推荐的风格通过返回一个对象做为模块对象，CommonJS的风格通过对module.exports或exports的属性赋值来达到暴露模块对象的目的。

CMD模块方式

define(function(require, exports, module) {

// 模块代码

});

**Javascript**垃圾回收方法

标记清除（mark and sweep）

这是JavaScript最常见的垃圾回收方式，当变量进入执行环境的时候，比如函数中声明一个变量，垃圾回收器将其标记为“进入环境”，当变量离开环境的时候（函数执行结束）将其标记为“离开环境”。

垃圾回收器会在运行的时候给存储在内存中的所有变量加上标记，然后去掉环境中的变量以及被环境中变量所引用的变量（闭包），在这些完成之后仍存在标记的就是要删除的变量了。

栈和队列的区别**?**

栈的插入和删除操作都是在一端进行的，而队列的操作却是在两端进行的。

队列先进先出，栈先进后出。

栈只允许在表尾一端进行插入和删除，而队列只允许在表尾一端进行插入，在表头一端进行删除

栈和堆的区别？

栈区（stack）— 由编译器自动分配释放 ，存放函数的参数值，局部变量的值等。

堆区（heap） — 一般由程序员分配释放， 若程序员不释放，程序结束时可能由OS回收。

堆（数据结构）：堆可以被看成是一棵树，如：堆排序；

栈（数据结构）：一种先进后出的数据结构。

谈谈你对**webpack**的看法

WebPack 是一个模块打包工具，你可以使用WebPack管理你的模块依赖，并编绎输出模块们所需的静态文件。它能够很好地管理、打包Web开发中所用到的HTML、Javascript、CSS以及各种静态文件（图片、字体等），让开发过程更加高效。对于不同类型的资源，webpack有对应的模块加载器。webpack模块打包器会分析模块间的依赖关系，最后 生成了优化且合并后的静态资源。

webpack的两大特色：

（1）.code splitting（可以自动完成）

（2）.loader 可以处理各种类型的静态文件，并且支持串联操作

webpack 是以commonJS的形式来书写脚本滴，但对 AMD/CMD 的支持也很全面，方便旧项目进行代码迁移。

webpack具有requireJs和browserify的功能，但仍有很多自己的新特性：

（1）. 对 CommonJS 、 AMD 、ES6的语法做了兼容

（2）. 对js、css、图片等资源文件都支持打包

（3）. 串联式模块加载器以及插件机制，让其具有更好的灵活性和扩展性，例如提供对CoffeeScript、ES6的支持

（4）. 有独立的配置文件webpack.config.js

（5）. 可以将代码切割成不同的chunk，实现按需加载，降低了初始化时间

（6）. 支持 SourceUrls 和 SourceMaps，易于调试

（7）. 具有强大的Plugin接口，大多是内部插件，使用起来比较灵活

（8）.webpack 使用异步 IO 并具有多级缓存。这使得 webpack 很快且在增量编译上更加快

**HTTP**和**HTTPS**

HTTP协议通常承载于TCP协议之上，在HTTP和TCP之间添加一个安全协议层（SSL或TSL），这个时候，就成了我们常说的HTTPS。

默认HTTP的端口号为80，HTTPS的端口号为443。

为什么HTTPS安全

因为网络请求需要中间有很多的服务器路由器的转发。中间的节点都可能篡改信息，而如果使用HTTPS，密钥在你和终点站才有。https之所以比http安全，是因为他利用ssl/tls协议传输。它包含证书，卸载，流量转发，负载均衡，页面适配，浏览器适配，refer传递等。保障了传输过程的安全性。

常见**web**安全及防护原理

sql注入原理

就是通过把SQL命令插入到Web表单递交或输入域名或页面请求的查询字符串，最终达到欺骗服务器执行恶意的SQL命令。

总的来说有以下几点：

（1）.永远不要信任用户的输入，要对用户的输入进行校验，可以通过正则表达式，或限制长度，对单引号和双"-"进行转换等。

（2）.永远不要使用动态拼装SQL，可以使用参数化的SQL或者直接使用存储过程进行数据查询存取。

（3）.永远不要使用管理员权限的数据库连接，为每个应用使用单独的权限有限的数据库连接。

（4）.不要把机密信息明文存放，请加密或者hash掉密码和敏感的信息。

21、javascript继承的6种方法

（1）原型链继承

（2）借用构造函数继承

（3）组合继承(原型+借用构造)

（4）原型式继承

（5）寄生式继承

（6），寄生组合式继承

**Node.js**的适用场景？

高并发、聊天、实时消息推送

说说严格模式的限制

变量必须声明后再使用

函数的参数不能有同名属性，否则报错

不能使用with语句

不能对只读属性赋值，否则报错

不能使用前缀0表示八进制数，否则报错

不能删除不可删除的属性，否则报错

不能删除变量delete prop，会报错，只能删除属性delete global[prop]

eval不会在它的外层作用域引入变量

eval和arguments不能被重新赋值

arguments不会自动反映函数参数的变化

不能使用arguments.callee

不能使用arguments.caller

禁止this指向全局对象

不能使用fn.caller和fn.arguments获取函数调用的堆栈

增加了保留字（比如protected、static和interface）

设立”严格模式”的目的，主要有以下几个：

消除Javascript语法的一些不合理、不严谨之处，减少一些怪异行为;

消除代码运行的一些不安全之处，保证代码运行的安全；

提高编译器效率，增加运行速度；

为未来新版本的Javascript做好铺垫。

注：经过测试IE6,7,8,9均不支持严格模式。

**new**操作符具体干了什么呢**?**

（1）、创建一个空对象，并且 this 变量引用该对象，同时还继承了该函数的原型。

（2）、属性和方法被加入到 this 引用的对象中。

（3）、新创建的对象由 this 所引用，并且最后隐式的返回 this 。

**jQuery** 中如何将数组转化为**json**字符串，然后再转化回来？

jQuery中没有提供这个功能，所以你需要先编写两个jQuery的扩展：

$.fn.stringifyArray = function(array) {

return JSON.stringify(array)

}

$.fn.parseArray = function(array) {

return JSON.parse(array)

}

然后调用：

$("").stringifyArray(array)

针对 **jQuery** 的优化方法？

\*基于Class的选择性的性能相对于Id选择器开销很大，因为需遍历所有DOM元素。

\*频繁操作的DOM，先缓存起来再操作。用Jquery的链式调用更好。

比如：var str=$("a").attr("href");

\*for (var i = size; i < arr.length; i++) {}

for 循环每一次循环都查找了数组 (arr) 的.length 属性，在开始循环的时候设置一个变量来存储这个数字，可以让循环跑得更快：

for (var i = size, length = arr.length; i < length; i++) {}

如何编写高性能的**Javascript**？

1使用 DocumentFragment 优化多次 append

2通过模板元素 clone ，替代 createElement

3使用 firstChild 和 nextSibling 代替 childNodes 遍历 dom 元素

4使用三目运算符替代条件分支

5需要不断执行的时候，优先考虑使用 setInterval

如何判断当前脚本运行在浏览器还是**node**环境中？

通过判断Global对象是否为window，如果不为window，当前脚本没有运行在浏览器中

对**Node**的优点和缺点提出了自己的看法？

\*（优点）因为Node是基于事件驱动和无阻塞的，所以非常适合处理并发请求，

因此构建在Node上的代理服务器相比其他技术实现（如Ruby）的服务器表现要好得多。

此外，与Node代理服务器交互的客户端代码是由javascript语言编写的，

因此客户端和服务器端都用同一种语言编写，这是非常美妙的事情。

\*（缺点）Node是一个相对新的开源项目，所以不太稳定，它总是一直在变，

而且缺少足够多的第三方库支持。看起来，就像是Ruby/Rails当年的样子。

平时如何管理你的项目？

先期团队必须确定好全局样式（globe.css），编码模式(utf-8) 等；

编写习惯必须一致（例如都是采用继承式的写法，单样式都写成一行）；

标注样式编写人，各模块都及时标注（标注关键样式调用的地方）；

页面进行标注（例如 页面 模块 开始和结束）；

CSS跟HTML 分文件夹并行存放，命名都得统一（例如style.css）；

JS 分文件夹存放 命名以该JS功能为准的英文翻译。

图片采用整合的 images.png png8 格式文件使用 尽量整合在一起使用方便将来的管理

说一下什么是**javascript**的同源策略？

同源策略是客户端脚本（尤其是Javascript）的重要的安全度量标准。它最早出自Netscape Navigator2.0，其目的是防止某个文档或脚本从多个不同源装载。

这里的同源策略指的是：协议，域名，端口相同，同源策略是一种安全协议。

指一段脚本只能读取来自同一来源的窗口和文档的属性。

为什么要有同源限制？

我们举例说明：比如一个黑客程序，他利用Iframe把真正的银行登录页面嵌到他的页面上，当你使用真实的用户名，密码登录时，他的页面就可以通过Javascript读取到你的表单中input中的内容，这样用户名，密码就轻松到手了。

缺点：

现在网站的JS 都会进行压缩，一些文件用了严格模式，而另一些没有。这时这些本来是严格模式的文件，被 merge 后，这个串就到了文件的中间，不仅没有指示严格模式，反而在压缩后浪费了字节。

说说最近最流行的一些东西吧？常去哪些网站？

Node.js、Mongodb、npm、MVVM、MEAN、three.js、vue.js、react。。。。。

你遇到过比较难的技术问题是？你是如何解决的？

常使用的库有哪些？常用的前端开发工具？开发过什么应用或组件？

列举**IE** 与其他浏览器不一样的特性？

什么叫优雅降级和渐进增强？

渐进增强 ：针对低版本浏览器进行构建页面，保证最基本的功能，然后再针对高级浏览器进行效果、交互等改进和追加功能达到更好的用户体验。

优雅降级 ：一开始就构建完整的功能，然后再针对低版本浏览器进行兼容。

除了前端以外还了解什么其它技术么？你最最厉害的技能是什么？

你常用的开发工具是什么，为什么？

对前端界面工程师这个职位是怎么样理解的？它的前景会怎么样？

前端是最贴近用户的程序员，比后端、数据库、产品经理、运营、安全都近。

1、实现界面交互

2、提升用户体验

3、有了Node.js，前端可以实现服务端的一些事情

前端是最贴近用户的程序员，前端的能力就是能让产品从 90分进化到 100 分，甚至更好，

参与项目，快速高质量完成实现效果图，精确到1px；

与团队成员，UI设计，产品经理的沟通；

做好的页面结构，页面重构和用户体验；

处理hack，兼容、写出优美的代码格式；

针对服务器的优化、拥抱最新前端技术。

你还有什么问题要问吗**?**

　　问目前在做的项目、使用的技术、晋升机制

**Angular**JS部分

**ng-repeat**迭代数组的时候，如果数组中有相同值，会有什么问题，如何解决？

会提示 Duplicates in a repeater are not allowed. 加 track by $index 可解决。当然，也可以 trace by 任何一个普通的值，只要能唯一性标识数组中的每一项即可（建立 dom 和数据之间的关联）。

**ng-click** 中写的表达式，能使用 **JS** 原生对象上的方法吗？

不止是 ng-click 中的表达式，只要是在页面中，都不能直接调用原生的 JS 方法，因为这些并不存在于与页面对应的 Controller 的 $scope 中。

举个栗子：

<p>{{parseInt(55.66)}}<p>

会发现，什么也没有显示。

但如果在 $scope 中添加了这个函数：

$scope.parseInt = function(x){

return parseInt(x);

}

这样自然是没什么问题了。

对于这种需求，使用一个 filter 或许是不错的选择：

<p>{{13.14 | parseIntFilter}}</p>

app.filter('parseIntFilter', function(){

return function(item){

return parseInt(item);

}

})

**angular** 的数据绑定采用什么机制？详述原理

脏检查机制。

双向数据绑定是 AngularJS 的核心机制之一。当 view 中有任何数据变化时，会更新到 model ，当 model 中数据有变化时，view 也会同步更新，显然，这需要一个监控。

原理就是，Angular 在 scope 模型上设置了一个 监听队列，用来监听数据变化并更新 view 。每次绑定一个东西到 view 上时 AngularJS 就会往 $watch 队列里插入一条 $watch，用来检测它监视的 model 里是否有变化的东西。当浏览器接收到可以被 angular context 处理的事件时，$digest 循环就会触发，遍历所有的 $watch，最后更新 dom。

举个栗子

<button ng-click="val=val+1">increase 1</button>

click 时会产生一次更新的操作（至少触发两次 $digest 循环）

·

按下按钮

浏览器接收到一个事件，进入到 angular context

$digest 循环开始执行，查询每个 $watch 是否变化

由于监视 $scope.val 的 $watch 报告了变化，因此强制再执行一次 $digest 循环

新的 $digest 循环未检测到变化

浏览器拿回控制器，更新 $scope.val 新值对应的 dom

$digest 循环的上限是 10 次（超过 10次后抛出一个异常，防止无限循环）。

两个平级界面块 **a** 和 **b**，如果 **a** 中触发一个事件，有哪些方式能让 **b** 知道？详述原理

这个问题换一种说法就是，如何在平级界面模块间进行通信。有两种方法，一种是共用服务，一种是基于事件。

共用服务

在 Angular 中，通过 factory 可以生成一个单例对象，在需要通信的模块 a 和 b 中注入这个对象即可。

基于事件

这个又分两种方式

第一种是借助父 controller。在子 controller 中向父 controller 触发（$emit）一个事件，然后在父 controller 中监听（$on）事件，再广播（$broadcast）给子 controller ，这样通过事件携带的参数，实现了数据经过父 controller，在同级 controller 之间传播。

第二种是借助 $rootScope。每个 Angular 应用默认有一个根作用域 $rootScope， 根作用域位于最顶层，从它往下挂着各级作用域。所以，如果子控制器直接使用 $rootScope 广播和接收事件，那么就可实现同级之间的通信。

逻辑代码的拆分

作为一个 MVVM 框架，Angular 应用本身就应该按照 模型，视图模型（控制器），视图来划分。

这里逻辑代码的拆分，主要是指尽量让 controller 这一层很薄。提取共用的逻辑到 service 中 （比如后台数据的请求，数据的共享和缓存，基于事件的模块间通信等），提取共用的界面操作到 directive 中（比如将日期选择、分页等封装成组件等），提取共用的格式化操作到 filter 中等等。

在复杂的应用中，也可以为实体建立对应的构造函数，比如硬盘（Disk）模块，可能有列表、新建、详情这样几个视图，并分别对应的有 controller，那么可以建一个 Disk 构造函数，里面完成数据的增删改查和验证操作，有跟 Disk 相关的 controller，就注入 Disk 构造器并生成一个实例，这个实例就具备了增删改查和验证方法。这样既层次分明，又实现了复用（让 controller 层更薄了）。

**angular** 应用常用哪些路由库，各自的区别是什么？

Angular1.x 中常用 ngRoute 和 ui.router，还有一种为 Angular2 设计的 new router（面向组件）。后面那个没在实际项目中用过，就不讲了。

无论是 ngRoute 还是 ui.router，作为框架额外的附加功能，都必须以 模块依赖 的形式被引入。

区别

ngRoute 模块是 Angular 自带的路由模块，而 ui.router 模块是基于 ngRoute模块开发的第三方模块。

ui.router 是基于 state （状态）的， ngRoute 是基于 url 的，ui.router模块具有更强大的功能，主要体现在视图的嵌套方面。

使用 ui.router 能够定义有明确父子关系的路由，并通过 ui-view 指令将子路由模版插入到父路由模板的 <div ui-view></div> 中去，从而实现视图嵌套。而在 ngRoute 中不能这样定义，如果同时在父子视图中 使用了 <div ng-view></div> 会陷入死循环。

示例

ngRoute

var app = angular.module('ngRouteApp', ['ngRoute']);

app.config(function($routeProvider){

$routeProvider

.when('/main', {

templateUrl: "main.html",

controller: 'MainCtrl'

})

.otherwise({ redirectTo: '/tabs' });

ui.router

var app = angular.module("uiRouteApp", ["ui.router"]);

app.config(function($urlRouterProvider, $stateProvider){

$urlRouterProvider.otherwise("/index");

$stateProvider

.state("Main", {

url: "/main",

templateUrl: "main.html",

controller: 'MainCtrl'

})

如何看待**angular2**

相比 Angular1.x，Angular2的改动很大，几乎算是一个全新的框架。

基于 TypeScript（可以使用 TypeScript 进行开发），在大型项目团队协作时，强语言类型更有利。

组件化，提升开发和维护的效率。

还有 module 支持动态加载，new router，promise的原生支持等等。

什么是**angular**的广播与事件

AngularJS 中在触发事件和发送广播时，都可以传递参数，可以通过这一特性，来实现数据的共享。与事件和广播相关的，共有三个方法，分别是：

    （1）.$emit()：触发事件，它可以向上传递数据，比如，子控制器向父控制器，还有控制器向 $rootScope

    （2）.$broadcast()：发送广播，它可以向下传递数据，比如，父控制器向子控制器传递数据，或者 $rootScope 向任意控制器传递数据

    （3）.$on()：监听事件与广播，可以捕获 $emit 和 $broadcast

可以将控制器之间的通信，分为三种情形：

    （1）.无直接关联的控制器：使用 $rootScope.$emit()、$rootScope.$boardcast() 或 $scope.$emit来发出数据，通过 $rootScope.$on() 来获取数据

    （2）.父控制器到子控制器：父控制器使用 $scope.$boradcast() 来发送数据，子控制器通过 $scope.$on() 来获取数据

（3）.子控制器至父控制器：子控制器使用 $scope.$emit() 来发送数据，父控制器通过 $scope.$on() 来获取数据

而且缺少足够多的第三方库支持。看起来，就像是Ruby/Rails当年的样子。

5、详述 angular 的 “依赖注入”

依赖注入是一种软件设计模式，目的是处理代码之间的依赖关系，减少组件间的耦合。AngularJS 是通过构造函数的参数名字来推断依赖服务名称的，通过 toString() 来找到这个定义的 function 对应的字符串，然后用正则解析出其中的参数（依赖项），再去依赖映射中取到对应的依赖，实例化之后传入6、ng-if 跟 ng-show/hide 的区别有哪些？

第一点区别是， ng-if 在后面表达式为 true 的时候才创建这个 dom 节点， ng-show 是初始时就创建了，用 display:block 和 display:none 来控制显示和不显示。

第二点区别是， ng-if 会（隐式地）产生新作用域， ng-switch 、 ng-include 等会动态创建一块界面的也是如此。

7、factory、service 和 provider 是什么关系？

把 service 的方法和数据放在一个对象里，并返回这个对象；通过构造函数方式创建 service，返回一个实例化对象；创建一个可通过 config 配置的 service，$get 中返回的，就是用 factory 创建 service 的内容

表达式 **{{yourModel}}**是如何工作的？

它依赖于 $interpolation服务，在初始化页面html后，它会找到这些表达式，并且进行标记，于是每遇见一个{{}}，则会设置一个$watch。而$interpolation会返回一个带有上下文参数的函数，最后该函数执行，则算是表达式$parse到那个作用域上。

列出至少三种实现不同模块之间通信方式？

Service

events,指定绑定的事件

使用 $rootScope

controller之间直接使用$parent, $$childHead等

directive 指定属性进行数据绑定

有哪些措施可以改善**Angular** 性能

官方提倡的，关闭debug,$compileProvider

myApp.config(function ($compileProvider) {

$compileProvider.debugInfoEnabled(false);

});

使用一次绑定表达式即{{::yourModel}}

减少watcher数量

在无限滚动加载中避免使用ng-repeat,关于解决方法可以参考这篇文章

使用性能测试的小工具去挖掘你的angular性能问题，我们可以使用简单的console.time()也可以借助开发者工具以及Batarang

**VUE**部分

<https://juejin.im/post/5ab2ff496fb9a028c06ab78f>

**React**部分

[https://zhuanlan.zhihu.com/p/24856035#tipjar](https://zhuanlan.zhihu.com/p/24856035%23tipjar)

其他补充

<https://juejin.im/post/59ffb4b66fb9a04512385402>