## 流行框架面试题

**1，常使用的库有哪些？**

\* 使用率较高的框架有jQuery、YUI、Prototype、Dojo、Ext.js、Mootools等。尤其是jQuery，超过91%。

轻量级框架有Modernizr、underscore.js、backbone.js、Raphael.js等。

**2，Node.js的适用场景？**

高并发、聊天、实时消息推送

**3.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。

**4 .Jquery与jQuery UI 有啥区别？**

\*jQuery是一个js库，主要提供的功能是选择器，属性修改和事件绑定等等。

\*jQuery UI则是在jQuery的基础上，利用jQuery的扩展性，设计的插件。

提供了一些常用的界面元素，诸如对话框、拖动行为、改变大小行为等等

**5.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)

**6，你说你热爱前端，那么应该WEB行业的发展很关注吧？ 说说最近最流行的一些东西吧？**

Node.js、Mongodb、npmM、MVVM、MEAN

**7，请写出jquery绑定事件的方法，不少于两种。**

.bind()是直接绑定在元素上.

.live()则是通过冒泡的方式来绑定到元素上的。更适合列表类型的，绑定到document DOM节点上。和.bind()的优势是支持动态数据。(live由于性能原因已经被废弃)

.delegate()则是更精确的小范围使用事件代理，性能优于.live()

.on()则是最新的1.9版本整合了之前的三种方式的新事件绑定机制

1.无论使用bind、on、delegate、click(function())都是重复绑定，即绑定的同类型事件被放到一个事件队列中，依次执行，后绑定的事件不会替换之前绑定的，对于on使用off，delegate用undelegate，bind及click使用unbind来解除绑定，例如unbind(type)传递为事件类型，如果不传type则解出所有事件绑定；需要注意的是元素本身自带的事件无法unbind（如button1）

2.要绑定自定义事件，如'open'，以上函数都可以使用，但激活需要使用trigger

3.建议使用on函数

$('.myClass').on({

click:function(eleDom){

...do someting

},

dbclick:function(eleDom){

...do someting

}

})

**8.请说出你可以传递给jQuery方法的四种不同值。**

选择器（字符串），HTML（字符串），回调函数，HTML元素，对象，数组，元素数组，jQuery对象等。

**9.什么是效果队列？**

jQuery中有个动画队列的机制。当我们对一个对象添加多次动画效果时后添加的动作就会被放入这个动画队列中，等前面的动画完成后再开始执行。可是用户的操作往往都比动画快，如果用户对一个对象频繁操作时不处理动画队列就会造成队列堆积，影响到效果。jQuery中有stop这个方法可以停止当前执行的动画，并且它有两个布尔参数，默认值都为false。第一个参数为true时会清空动画队列，第二个参数为true时会瞬间完成掉当前动画。所以，我们经常使用obj.stop(true,true)来停止动画。但是这还不够！正如jQuery文档中的说法，即使第二个参数为true，也仅仅是把当前在执行的动画跳转到完成状态。这时第一个参数如果也为true，后面的队列就会被清空。如果一个效果需要多个动画同时处理，我们仅完成其中的一个而把后面的队列丢弃了，这就会出现意料之外的结果。

**10.请指出.get()，[]，eq()的区别。**

eq:返回是一个jquery对象作用是将匹配的元素集合缩减为一个元素。这个元素在匹配元素集合中的位置变为0，而集合长度变成1。

get:是一个html对象数组作用是取得其中一个匹配的元素。num表示取得第几个匹配的元素。

85.请指出.bind()，.live()和.delegate()的区别。

在操纵DOM的语境中，document是根节点。现在我们可以较容易地说明.bind()、.live()和.delegate()的不同之处了。

.bind()

1 $('a').bind('click', function() {alert("That tickles!")});

这是最简单的绑定方法了。JQuery扫描文档找出所有的$(‘a’)元素，并把alert函数绑定到每个元素的click事件上。

.live()

1 $('a').live('click', function() {alert("That tickles!")});

JQuery把alert函数绑定到$(document)元素上，并使用’click’和’a’作为参数。任何时候只要有事件冒泡到document节点上，它就查看该事件是否是一个click事件，以及该事件的目标元素与’a’这一CSS选择器是否匹配，如果都是的话，则执行函数。

live方法还可以被绑定到具体的元素(或context)而不是document上，像这样：

1 $('a', $('#container')[0]).live(...);

.delegate()

1 $('#container').delegate('a', 'click', function() {alert("That tickles!")});

JQuery扫描文档查找$(‘#container’)，并使用click事件和’a’这一CSS选择器作为参数把alert函数绑定到$(‘#container’)上。任何时候只要有事件冒泡到$(‘#container’)上，它就查看该事件是否是click事件，以及该事件的目标元素是否与CCS选择器相匹配。如果两种检查的结果都为真的话，它就执行函数。

可以注意到，这一过程与.live()类似，但是其把处理程序绑定到具体的元素而非document这一根上。精明的JS’er们可能会做出这样的结论，即$('a').live()==$(document).delegate('a')，是这样吗?嗯，不，不完全是。

为什么.delegate()要比.live()好用？

基于几个原因，人们通常更愿意选用jQuery的delegate方法而不是live方法。考虑下面的例子：

1 $('a').live('click', function(){blah()}); // 或者

2 $(document).delegate('a', 'click', function(){blah()});

速度

后者实际上要快过前者，因为前者首先要扫描整个的文档查找所有的$(‘a’)元素，把它们存成jQuery对象。尽管live函数仅需要把’a’作为串参数传递以用做之后的判断，但是$()函数并未知道被链接的方法将会是.live()。而另一方面，delegate方法仅需要查找并存储$(document)元素。

一种寻求避开这一问题的方法是调用在$(document).ready()之外绑定的live，这样它就会立即执行。在这种方式下，其会在DOM获得填充之前运行，因此就不会查找元素或是创建jQuery对象了。

灵活性和链能力

live函数也挺令人费解的。想想看，它被链到$(‘a’)对象集上，但其实际上是在$(document)对象上发生作用。由于这个原因，它能够试图以一种吓死人的方式来把方法链到自身上。实际上，我想说的是，以$.live(‘a’,…)这一形式作为一种全局性的jQuery方法，live方法会更具意义一些。

仅支持CSS选择器

最后一点，live方法有一个非常大的缺点，那就是它仅能针对直接的CSS选择器做操作，这使得它变得非常的不灵活。

为什么选择.live()或.delegate()而不是.bind()？

毕竟，bind看起来似乎更加的明确和直接，难道不是吗?嗯，有两个原因让我们更愿意选择delegate或live而不是bind：

为了把处理程序附加到可能还未存在于DOM中的DOM元素之上。因为bind是直接把处理程序绑定到各个元素上，它不能把处理程序绑定到还未存在于页面中的元素之上。

如果你运行了$(‘a’).bind(…)，而后新的链接经由AJAX加入到了页面中，则你的bind处理程序对于这些新加入的链接来说是无效的。而另一方面live和delegate则是被绑定到另一个祖先节点上，因此其对于任何目前或是将来存在于该祖先元素之内的元素都是有效的。

或者为了把处理程序附加到单个元素上或是一小组元素之上，监听后代元素上的事件而不是循环遍历并把同一个函数逐个附加到DOM中的100个元素上。把处理程序附加到一个(或是一小组)祖先元素上而不是直接把处理程序附加到页面中的所有元素上，这种做法带来了性能上的好处。

停止传播

最后一个我想做的提醒与事件传播有关。通常情况下，我们可以通过使用这样的事件方法来终止处理函数的执行：

1 $('a').bind('click', function(e) {

2 e.preventDefault();

3 // 或者 e.stopPropagation();

4 });

不过，当我们使用live或是delegate方法的时候，处理函数实际上并没有在运行，需要等到事件冒泡到处理程序实际绑定的元素上时函数才会运行。而到此时为止，我们的其他的来自.bind()的处理函数早已运行了。

**11.请指出$和$.fn的区别，或者说出$.fn的用途。**

Jquery为开发插件提供了两个方法，分别是：

1 $.extend(obj);

2 $.fn.extend(obj);

1.那么这两个分别是什么意思？

$.extend(obj);是为了扩展jquery本身，为类添加新的方法。

$.fn.extend(obj);给JQUERY对象添加方法。

2.$.fn中的fn是什么意思，其实是prototype，即$.fn=$.prototype;

具体用法请看下面的例子：

1 $.extend({

2

3 add:function(a, b) {

4 return a+b;

5 }

6 })

7

8 $.add(5,8); // return 13

注意没有，这边的调用直接调用，前面不用任何对象。直接$.+方法名

$.fn.extend(obj)；对prototype进行扩展，为jquery类添加成员函数，jquery类的实例可以使用这个成员函数。

1 $.fn.extend({

2

3 clickwhile:function(){

4

5 $(this).click(function(){

6 alert($(this).val())

7 })

8 }

9 })

10

11 $('input').clickwhile(); // 当点击输入框会弹出该对象的Value值

注意调用时候前面是有对象的。即$('input')这么个东西。

**12.请写出一个函数实现N!的计算。N取很大的值时，该怎么办？**

使用循环、递归都能写出函数。

当N取值很大时，应该考虑把数值转化为字符串再进行运算。大数乘法再转化为大数加法运算，其具体算法应该有不少C语言实现，可以参考一下。

1. **modulo(12,5) //2 ；问题：实现满足上述结果的modulo函数。**

**14."i'm a lasagna hog".split("").reverse().join("");问题：上面的语句的返回值是什么？**

答案："goh angasal a m'i";

**15.(window.foo||(window.foo="bar"));问题：window.foo的值是什么？**

答案："bar"只有window.foo为假时的才是上面答案，否则就是它本身的值。

var foo="Hello";(function(){var bar="World";alert(foo+bar);})();alert(foo+bar);

1. **问题：上面两个alert的结果是什么？**

答案:"Hello World"和ReferenceError:bar is not defined

**17..var foo=[];foo.push(1);foo.push(2);问题：foo.length的值是什么？**

答案：2

有趣的问题：

1. **你编写过的最酷的代码是什么？其中你最自豪的是什么？**
2. **在你使用过的开发工具中，最喜欢哪个？**
3. **你有什么业余项目吗？是哪种类型的？**

**21.你最爱的IE特性是什么？**

**22.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。

这样会导致，在 ng-if 中用基本变量绑定 ng-model，并在外层 div 中把此 model 绑定给另一个显示区域，内层改变时，外层不会同步改变，因为此时已经是两个变量了。

<p>{{name}}</p>

<div ng-if="true">

<input type="text" ng-model="name">

</div>

ng-show 不存在此问题，因为它不自带一级作用域。

避免这类问题出现的办法是，始终将页面中的元素绑定到对象的属性（data.x）而不是直接绑定到基本变量（x）上。

详见 [AngularJS 中的作用域](http://huangtengfei.com/2015/09/scope-in-angularjs/)

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);

}

})

{{now | 'yyyy-MM-dd'}} 这种表达式里面，竖线和后面的参数通过什么方式可以自定义？

filter，格式化数据，接收一个输入，按某规则处理，返回处理结果。

内置 filter

ng 内置的 filter 有九种：

date（日期）

currency（货币）

limitTo（限制数组或字符串长度）

orderBy（排序）

lowercase（小写）

uppercase（大写）

number（格式化数字，加上千位分隔符，并接收参数限定小数点位数）

filter（处理一个数组，过滤出含有某个子串的元素）

json（格式化 json 对象）

filter 有两种使用方法，一种是直接在页面里：

<p>{{now | date : 'yyyy-MM-dd'}}</p>

另一种是在 js 里面用：

// $filter('过滤器名称')(需要过滤的对象, 参数1, 参数2,...)

$filter('date')(now, 'yyyy-MM-dd hh:mm:ss');

自定义 filter

// 形式

app.filter('过滤器名称',function(){

return function(需要过滤的对象,过滤器参数1,过滤器参数2,...){

//...做一些事情

return 处理后的对象;

}

});

// 栗子

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

return function(item, times){

var result = '';

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

result += item;

}

return result;

}

})

**23.factory、service 和 provider 是什么关系？**

factory

把 service 的方法和数据放在一个对象里，并返回这个对象

app.factory('FooService', function(){

return {

target: 'factory',

sayHello: function(){

return 'hello ' + this.target;

}

}

});

service

通过构造函数方式创建 service，返回一个实例化对象

app.service('FooService', function(){

var self = this;

this.target = 'service';

this.sayHello = function(){

return 'hello ' + self.target;

}

});

provider

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

app.provider('FooService', function(){

this.configData = 'init data';

this.setConfigData = function(data){

if(data){

this.configData = data;

}

}

this.$get = function(){

var self = this;

return {

target: 'provider',

sayHello: function(){

return self.configData + ' hello ' + this.target;

}

}

}

});

// 此处注入的是 FooService 的 provider

app.config(function(FooServiceProvider){

FooServiceProvider.setConfigData('config data');

});

从底层实现上来看，service 调用了 factory，返回其实例；factory 调用了 provider，返回其 $get 中定义的内容。factory 和 service 功能类似，只不过 factory 是普通 function，可以返回任何东西（return 的都可以被访问，所以那些私有变量怎么写，你懂的）；service 是构造器，可以不返回（绑定到 this 的都可以被访问）；provider 是加强版 factory，返回一个可配置的 factory。

详见 [AngularJS 之 Factory vs Service vs Provider](http://www.oschina.net/translate/angularjs-factory-vs-service-vs-provider)

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）。

详见 [关于 AngularJS 的数据绑定](http://huangtengfei.com/2015/09/data-bind-of-angularjs/)

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。

详见 [AngularJS 中 Controller 之间的通信](http://huangtengfei.com/2015/09/communication-in-controller/)

**24.一个 angular 应用应当如何良好地分层？**

目录结构的划分

对于小型项目，可以按照文件类型组织，比如：

css

js

controllers

models

services

filters

templates

但是对于规模较大的项目，最好按业务模块划分，比如：

css

modules

account

controllers

models

services

filters

templates

disk

controllers

models

services

filters

templates

modules 下最好再有一个 common 目录来存放公共的东西。

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）。

参考 [AngularJS在苏宁云中心的深入实践](http://www.csdn.net/article/2015-08-24/2825527)

**26.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](https://angular.github.io/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'

})

如果通过angular的directive规划一套全组件化体系，可能遇到哪些挑战？

没有自己用 directive 做过一全套组件，讲不出。

能想到的一点是，组件如何与外界进行数据的交互，以及如何通过简单的配置就能使用吧。

分属不同团队进行开发的 angular 应用，如果要做整合，可能会遇到哪些问题，如何解决？

可能会遇到不同模块之间的冲突。

比如一个团队所有的开发在 moduleA 下进行，另一团队开发的代码在 moduleB 下

angular.module('myApp.moduleA', [])

.factory('serviceA', function(){

...

})

angular.module('myApp.moduleB', [])

.factory('serviceA', function(){

...

})

angular.module('myApp', ['myApp.moduleA', 'myApp.moduleB'])

会导致两个 module 下面的 serviceA 发生了覆盖。

貌似在 Angular1.x 中并没有很好的解决办法，所以最好在前期进行统一规划，做好约定，严格按照约定开发，每个开发人员只写特定区块代码。

**27.angular 的缺点有哪些？**

强约束

导致学习成本较高，对前端不友好。

但遵守 AngularJS 的约定时，生产力会很高，对 Java 程序员友好。

不利于 SEO

因为所有内容都是动态获取并渲染生成的，搜索引擎没法爬取。

一种解决办法是，对于正常用户的访问，服务器响应 AngularJS 应用的内容；对于搜索引擎的访问，则响应专门针对 SEO 的HTML页面。

性能问题

作为 MVVM 框架，因为实现了数据的双向绑定，对于大数组、复杂对象会存在性能问题。

可以用来 [优化 Angular 应用的性能](https://github.com/xufei/blog/issues/23) 的办法：

减少监控项（比如对不会变化的数据采用单向绑定）

主动设置索引（指定 track by，简单类型默认用自身当索引，对象默认使用 $$hashKey，比如改为 track by item.id）

降低渲染数据量（比如分页，或者每次取一小部分数据，根据需要再取）

数据扁平化（比如对于树状结构，使用扁平化结构，构建一个 map 和树状数据，对树操作时，由于跟扁平数据同一引用，树状数据变更会同步到原始的扁平数据）

另外，对于Angular1.x ，存在 脏检查 和 模块机制 的问题。

移动端

可尝试 Ionic，但并不完善。

参考

1. **[如何看2015年1月Peter-Paul Koch对Angular的看法？](https://www.zhihu.com/question/27601964/answer/37309005)**

**29.如何看待 angular 1.2 中引入的 controller as 语法？**

最根本的好处

在 angular 1.2 以前，在 view 上的任何绑定都是直接绑定在 $scope 上的

function myCtrl($scope){

$scope.a = 'aaa';

$scope.foo = function(){

...

}

}

使用 controllerAs，不需要再注入 $scope，controller 变成了一个很简单的 javascript 对象（POJO），一个更纯粹的 ViewModel。

function myCtrl(){

// 使用 vm 捕获 this 可避免内部的函数在使用 this 时导致上下文改变

var vm = this;

vm.a = 'aaa';

}

原理

从源码实现上来看，controllerAs 语法只是把 controller 这个对象的实例用 as 别名在 $scope 上创建了一个属性。

if (directive.controllerAs) {

locals.$scope[directive.controllerAs] = controllerInstance;

}

但是这样做，除了上面提到的使 controller 更加 POJO 外，还可以避免遇到 AngularJS 作用域相关的一个坑（就是上文中 ng-if 产生一级作用域的坑，其实也是 javascript 原型链继承中值类型继承的坑。因为使用 controllerAs 的话 view 上所有字段都绑定在一个引用的属性上，比如 vm.xx，所以坑不再存在）。

<div ng-controller="TestCtrl as vm">

<p>{{name}}</p>

<div ng-if="vm.name">

<input type="text" ng-model="vm.name">

</div>

</div>

问题

使用 controllerAs 会遇到的一个问题是，因为没有注入 $scope，导致 $emit、 $broadcast、 $on、 $watch 等 $scope 下的方法无法使用。这些跟事件相关的操作可以封装起来统一处理，或者在单个 controller 中引入 $scope，特殊对待。

参考 [angular controller as syntax vs scope](http://www.cnblogs.com/whitewolf/p/3493362.html)

详述 angular 的 “依赖注入”

栗子

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。

举个栗子，如果没有使用 AngularJS，想从后台查询数据并在前端显示，可能需要这样做：

var animalBox = document.querySelector('.animal-box');

var httpRequest = {

get: function(url, callback){

console.log(url + ' requested');

var animals = ['cat', 'dog', 'rabbit'];

callback(animals);

}

}

var render = function(el, http){

http.get('/api/animals', function(animals){

el.innerHTML = animals;

})

}

render(httpRequest, animalBox);

但是，如果在调用 render 的时候不传参数，像下面这样，会报错，因为找不到 el 和 http（定义的时候依赖了，运行的时候不会自动查找依赖项）

render();

// TypeError: Cannot read property 'get' of undefined

而使用 AngularJS，可以直接这样

function myCtrl = ($scope, $http){

$http.get('/api/animals').success(function(data){

$scope.animals = data;

})

}

也就是说，在 Angular App 运行的时候，调用 myCtrl，自动做了 $scope 和 $http 两个依赖性的注入。

原理

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。

简化一下，大概是这样：

var inject = {

// 存储依赖映射关系

storage: {},

// 注册依赖

register: function(name, resource){

this.storage[name] = resource;

},

// 解析出依赖并调用

resolve: function(target){

var self = this;

var FN\_ARGS = /^function\s\*[^\(]\*\(\s\*([^\)]\*)\)/m;

var STRIP\_COMMENTS = /((\/\/.\*$)|(\/\\*[\s\S]\*?\\*\/))/mg;

fnText = target.toString().replace(STRIP\_COMMENTS, '');

argDecl = fnText.match(FN\_ARGS)[1].split(/, ?/g);

var args = [];

argDecl.forEach(function(arg){

if(self.storage[arg]){

args.push(self.storage[arg]);

}

})

return function(){

target.apply({}, args);

}

}

}

使用这个 injector，前面那个不用 AngularJS 的栗子这样改造一下就可以调用了

inject.register('el', animalBox);

inject.register('ajax', httpRequest);

render = inject.resolve(render);

render();

问题

因为 AngularJS 的 injector 是假设函数的参数名就是依赖的名字，然后去查找依赖项，那如果按前面栗子中那样注入依赖，代码压缩后（参数被重命名了），就无法查找到依赖项了。

// 压缩前

function myCtrl = ($scope, $http){

...

}

// 压缩后

function myCtrl = (a, b){

...

}

所以，通常会使用下面两种方式注入依赖（对依赖添加的顺序有要求）。

数组注释法

myApp.controller('myCtrl', ['$scope', '$http', function($scope, $http){

...

}])

显式 $inject

myApp.controller('myCtrl', myCtrl);

function myCtrl = ($scope, $http){

...

}

myCtrl.$inject = ['$scope', '$http'];

补充

对于一个 DI 容器，必须具备三个要素：依赖项的注册，依赖关系的声明和对象的获取。

在 AngularJS 中，module 和 $provide 都可以提供依赖项的注册；内置的 injector 可以获取对象（自动完成依赖注入）；依赖关系的声明，就是前面问题中提到的那样。

下面是个栗子

// 对于 module，传递参数不止一个，代表新建模块，空数组代表不依赖其他模块

// 只有一个参数（模块名），代表获取模块

// 定义 myApp，添加 myApp.services 为其依赖项

angular.module('myApp', ['myApp.services']);

// 定义一个 services module，将 services 都注册在这个 module 下面

angular.module('myApp.services', [])

// $provider 有 factory, service, provider, value, constant

// 定义一个 HttpService

angular.module('myApp.services').service('HttpService', ['$http', function($http){

...

}])

参考

[[AngularJS] 自己实现一个简单的依赖注入](https://segmentfault.com/a/1190000003048407cope-in-angularjs/)

[理解angular中的module和injector，即依赖注入](http://www.mamicode.com/info-detail-247448.html)

[AngularJS中的依赖注入实际应用场景](https://www.zhihu.com/question/28097646)

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。

迎合未来标准，吸纳其他框架的优点，值得期待，不过同时要学习的东西也更多了（ES next、TS、Rx等）。

**30. angular的数据绑定采用什么机制？详述原理**

angularjs的双向数据绑定,采用脏检查（dirty-checking）机制。ng只有在指定事件触发后，才进入 $digest cycle ：

- DOM事件，譬如用户输入文本，点击按钮等。( ng-click )

- XHR响应事件 ( $http )

- 浏览器Location变更事件 ( $location )

- Timer事件(

timeout,

timeout,

interval )

- 执行

digest()或

digest()或

apply()

2. 依赖注入(DI)

让我们可以不用自己实例化就能创建依赖对象的方法. 简单的来说, 依赖是以注入的方式传递的。在Web应用中， Angular让我们可以通过DI来创建像Controllers和Directives这样的对象. 我们还可以创建自己的依赖对象, 当我们要实例化它们时, Angular能自动实现注入。

3. compile和link的区别：性能和能力

编译的时候，compile转换dom，碰到绑定监听器的地方就先存着，有几个存几个，到最后汇总成一个link函数，一并执行，提升了性能。

|  |
| --- |
| function compile(tElement, tAttrs, transclude) { ... }tElement为编译前的element  function link(scope, iElement, iAttrs, controller) { ... } iElement为编译后的element，已经与作用域关联起来，所以可以数据绑定 |

如果指令只进行DOM的修改，不进行数据绑定，那么配置在compile函数中，如果指令要进行数据绑定，那么配置在link函数中。

4.

apply()和

digest()的区别

安全性：

apply()可以接收一个参数作为function()，这个function会被包装到一个try…catch块中，所以一旦有异常发生，该异常会被

apply()可以接收一个参数作为function()，这个function会被包装到一个try…catch块中，所以一旦有异常发生，该异常会被

exceptionHandler service 处理。

apply会使ng进入 apply会使ng进入 digest cycle , 并从$rootScope开始遍历(深度优先)检查数据变更。

digest仅会检查该scope和它的子scope，当你确定当前操作仅影响它们时，用 digest仅会检查该scope和它的子scope，当你确定当前操作仅影响它们时，用 digest可以稍微提升性能。

5.ng-if 跟 ng-show/hide 的区别有哪些？

第一点区别：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：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。

这样会导致，在 ng-if 中用基本变量绑定 ng-model，并在外层 div 中把此 model 绑定给另一个显示区域，内层改变时，外层不会同步改变，因为此时已经是两个变量了。

<p>{{name}}</p>

<div ng-if="true">

<input type="text" ng-model="name">

</div>

ng-show 不存在此问题，因为它不自带一级作用域。

避免这类问题出现的办法是，始终将页面中的元素绑定到对象的属性（data.x）而不是直接绑定到基本变量（x）上。

**31.什么是错误优先的回调函数？**

**32.如何避免回调地狱？**

**33.如何用Node来监听80端口？**

**34.什么是事件循环？**

**35.哪些工具可以用来保证一致的编程风格？**

**36.什么是测试金字塔？对于HTTP API，如何利用测试金字塔？**

**37.你最喜欢的HTTP框架，并说明原因？**

**38.现在，我们依次来解答这些问题吧。**

**39.什么是错误优先的回调函数？**

错误优先的回调函数用于传递错误和数据。

第一个参数始终应该是一个错误对象， 用于检查程序是否发生了错误。其余的参数用于传递数据。例如：

Crayon Syntax Highlighter v2.7.1.1

JavaScript

|  |  |
| --- | --- |
| 1  2  3  4  5 | fs.readFile(filePath, function(err, data) {  if (err) {  //handle the error  }  // use the data |

[Format Time: 0.0011 seconds]

解析：这个题目的主要作用在于检查被面试者对于Node中异步操作的一些基本知识的掌握。

如何避免回调地狱

你可以有如下几个方法：

模块化：将回调函数分割为独立的函数

使用Promises

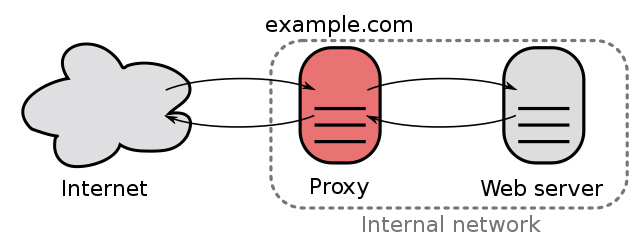
使用yield来计算生成器或Promise

解析：这个问题有很多种答案，取决你使用的场景，例如ES6, ES7，或者一些控制流库。

如何用Node监听80端口

这题有陷阱！在类Unix系统中你不应该尝试去监听80端口，因为这需要超级用户权限。 因此不推荐让你的应用直接监听这个端口。

目前，如果你一定要让你的应用监听80端口的话，你可以有通过在Node应用的前方再增加一层反向代理 （例如[nginx](http://nginx.org/)）来实现，如下图所示。否则，建议你直接监听大于1024的端口。



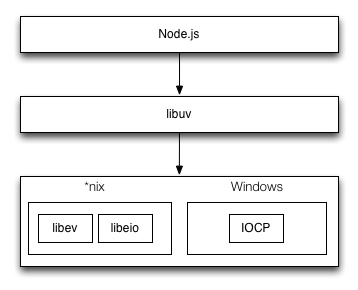
方向代理指的是以代理服务器来接收Internet上的连接请求，然后将请求转发给内部网络上的服务器， 并且将服务器返回的结果发送给客户端。

关于反向代理的更多内容，建议你阅读[这篇文章](http://www.cnblogs.com/edisonchou/p/4126742.html)。

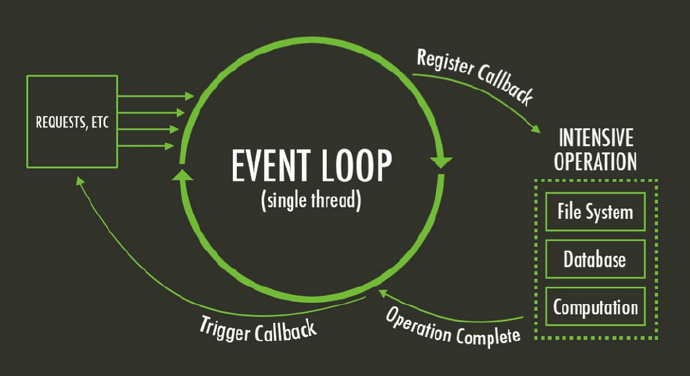
解释：这个问题用于检查被面试者是否有实际运行Node应用的经验。

什么是事件循环

Node采用的是单线程的处理机制（所有的I/O请求都采用非阻塞的工作方式），至少从Node.js开发者的角度是这样的。 而在底层，Node.js借助[libuv](https://github.com/libuv/libuv)来作为抽象封装层， 从而屏蔽不同操作系统的差异，Node可以借助livuv来来实现多线程。下图表示了Node和libuv的关系。



Libuv库负责Node API的执行。它将不同的任务分配给不同的线程，形成一个事件循环， 以异步的方式将任务的执行结果返回给V8引擎。可以简单用下面这张图来表示。



每一个I/O都需要一个回调函数——一旦执行完便推到事件循环上用于执行。 如果你需要更多详细的解释，可以参考[这个视频](https://www.youtube.com/embed/8aGhZQkoFbQ)。 你也可以参考[这篇文章](http://www.ruanyifeng.com/blog/2014/10/event-loop.html)。

解释：这用于检查Node.js的底层知识，例如什么是libuv，它的作用是什么。

哪些工具可以用来保证一致性的代码风格

你可以选择如下的工具：

[JSLint](http://jslint.com/)

[JSHint](http://jshint.com/)

[ESLint](http://eslint.org/)

[JSCS](http://jscs.info/) – 推荐

在团队开发中，这些工具对于编写代码非常的有帮助，能够帮助团队开发者强制执行规定的风格指南， 还能够通过静态分析捕获常见的错误。

解析：用于检查被面试者是否有大型项目开发经验。

运算错误与程序员错误的区别

运算错误并不是bug，这是和系统相关的问题，例如请求超时或者硬件故障。而程序员错误就是所谓的bug。

解析：这个题目和Node关系并不大，用于考察面试者的基础知识。

使用NPM有哪些好处？

通过NPM，你可以安装和管理项目的依赖，并且能够指明依赖项的具体版本号。 对于Node应用开发而言，你可以通过package.json文件来管理项目信息，配置脚本， 以及指明项目依赖的具体版本。

关于NPM的更多信息，你可以参考[官方文档](https://docs.npmjs.com/files/package.json)。

解析：它能考察面试者使用npm命令的基础知识和Node.js开发的实际经验。

什么是Stub？举个使用场景

Stub是用于模拟一个组件或模块的函数或程序。在测试用例中， 简单的说，你可以用Stub去模拟一个方法，从而避免调用真实的方法， 使用Stub你还可以返回虚构的结果。你可以配合断言使用Stub。

举个例子，在一个读取文件的场景中，当你不想读取一个真正的文件时：

Crayon Syntax Highlighter v2.7.1.1

JavaScript

[Format Time: 0.0017 seconds]

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | var fs = require('fs');    var readFileStub = sinon.stub(fs, 'readFile', function (path, cb) {  return cb(null, 'filecontent');  });    expect(readFileStub).to.be.called;  readFileStub. |

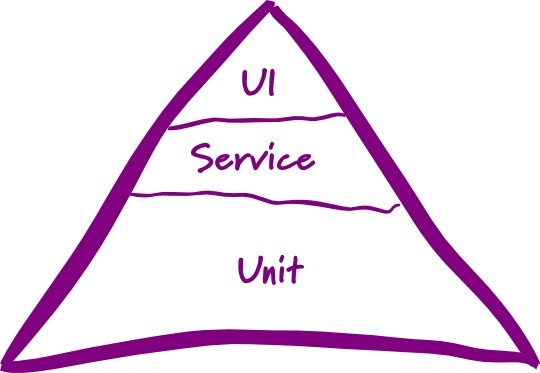
在单元测试中：Stub是完全模拟一个外部依赖，而Mock常用来判断测试通过还是失败。

有关Node.js的单元测试小结，你可以参考[这个链接](http://segmentfault.com/a/1190000002921481)。

解析：用于测试被面试者是否有测试的经验。如果被面试者知道什么是Stub， 那么可以继续问他是如何做单元测试的。

什么是测试金字塔？

[测试金字塔](http://zyzhang.github.io/blog/2013/04/28/test-pyramid/)指的是： 当我们在编写测试用例时，底层的单元测试应该远比上层的端到端测试要多。



当我们谈到HTTP API时，我们可能会涉及到：

有很多针对模型的底层单元测试

但你需要测试模型间如何交互时，需要减少集成测试

解析：本文主要考察被面试者的在测试方面的经验

**40、jQuery 的 slideUp动画 ，如果目标元素是被外部事件驱动, 当鼠标快速地连续触发外部元素事件, 动画会滞后的反复执行，该如何处理呢?**

先stop(true,true)后slideUp()

## JQuery一个对象可以同时绑定多个事件，这是如何实现的？

jQuery可以给一个对象同时绑定多个事件，低层实现方式是使用addEventListner或attachEvent兼容不同的浏览器实现事件的绑定，这样可以给同一个对象注册多个事件。

## 2、模块化开发怎么做？

理解模块化开发模式：浏览器端requirejs，seajs；服务器端nodejs；ES6模块化；fis、webpack等前端整体模块化解决方案；grunt、gulp等前端工作流的使用

## 3、AMD（Modules/Asynchronous-Definition）、CMD（Common Module Definition）规范区别？

理解这两种规范的差异，主要通过requirejs与seajs的对比，理解模块的定义与引用方式

的差异以及这两种规范的设计原则

参考链接1：<https://www.zhihu.com/question/20351507/answer/14859415>

参考链接2：<https://github.com/seajs/seajs/issues/277>

1、对于依赖的模块，AMD 是提前执行，CMD 是延迟执行。不过 RequireJS 从 2.0 开始，也改成可以延迟执行（根据写法不同，处理方式不同）。CMD 推崇 as lazy as possible.

2、CMD 推崇依赖就近，AMD 推崇依赖前置。

3. AMD 的 API 默认是一个当多个用，CMD 的 API 严格区分，推崇职责单一。比如 AMD 里，require 分全局 require 和局部 require，都叫 require。CMD 里，没有全局 require，而是根据模块系统的完备性，提供 seajs.use 来实现模块系统的加载启动。CMD 里，每个 API 都简单纯粹。

## 4、requireJS的核心原理是什么？（如何动态加载的？如何避免多次加载的？如何 缓存的？）

核心是js的加载模块，通过正则匹配模块以及模块的依赖关系，保证文件加载的先后顺序，根据文件的路径对加载过的文件做了缓存

## 5、让你自己设计实现一个requireJS，你会怎么做？

核心是实现js的加载模块，维护js的依赖关系，控制好文件加载的先后顺序

# 八、流行框架

## 1、JQuery的源码看过吗？能不能简单概况一下它的实现原理？

考察学习知识的态度，是否仅仅是停留在使用层面，要知其然知其所以然

## 2、jQuery.fn的init方法返回的this指的是什么对象？为什么要返回this？

this执行init构造函数自身，其实就是jQuery实例对象，返回this是为了实现jQuery的链式操作

## jquery中如何将数组转化为json字符串，然后再转化回来？

$.parseJSON('{"name":"John"}');

JSON.stringify

## jQuery 的属性拷贝(extend)的实现原理是什么，如何实现深拷贝？

递归赋值

## jquery.extend 与 jquery.fn.extend的区别？

Jquery.extend用来扩展jQuery对象本身；jquery.fn.extend用来扩展jQuery实例

## 6、谈一下Jquery中的bind(),live(),delegate(),on()的区别？

jquery1.7以后就推荐使用on的方式来进行事件绑定了

## 7、JQuery一个对象可以同时绑定多个事件，这是如何实现的？

可以同时绑定多个事件，低层实现原理是使用addEventListner与attachEvent兼容处理做事件注册

## Jquery与jQuery UI有啥区别？

jQuery是操作dom的框架，jQueryUI是基于jQuery做的一个UI组件库

## jQuery和Zepto的区别？各自的使用场景？

jQuery主要用于pc端，当然有对应的jQuerymobile用于移动端，zepto比jQuery更加小巧，主要用于移动端

jquer mobile相对于zepto功能强大，但是体积也很庞大，zepto非常的轻量

# 6、针对 jQuery 的优化方法？

a、优先使用ID选择器

b、jquery获取到的DOM元素如果需要多次使用，建议使用一个变量将其保存起来，因为操作DOM的过程是非常耗费性能的

c、在class前使用tag(标签名)

d、给选择器一个上下文

e、慎用 .live()方法（应该说尽量不要使用）

f、使用data()方法存储临时变量

## 7、Zepto的点透问题如何解决？

点透主要是由于两个div重合，例如：一个div调用show()，一个div调用hide()；这个时候当点击上面的div的时候就会影响到下面的那个div；

解决办法主要有2种：

1.github上有一个叫做fastclick的库，它也能规避移动设备上click事件的延迟响应，https://github.com/ftlabs/fastclick

将它用script标签引入页面（该库支持AMD，于是你也可以按照AMD规范，用诸如require.js的模块加载器引入），并且在dom ready时初始化在body上，

2.根据分析，如果不引入其它类库，也不想自己按照上述fastclcik的思路再开发一套东西，需要1.一个优先于下面的“divClickUnder”捕获的事件；2.并且通过这个事件阻止掉默认行为（下面的“divClickUnder”对click事件的捕获，在ios的safari，click的捕获被认为和滚屏、点击输入框弹起键盘等一样，是一种浏览器默认行为，即可以被event.preventDefault()阻止的行为）。

## 12、知道各种JS框架(Angular, Backbone, Ember, React, Meteor, Knockout...)么? 能讲出他们各自的优点和缺点么?

知识面的宽度，流行框架要多多熟悉

angular、backbone、knockout都是完整的MV\*框架

angular是双向数据绑定的，backbone、knockout是单向数据绑定的

React只是单纯地View层

## 13、Underscore 对哪些 JS 原生对象进行了扩展以及提供了哪些好用的函数方法？

Underscore的熟悉程度

## 14、使用过angular吗？angular中的过滤器是干什么用的

在表达式中转换数据<p>姓名为 {{ lastName | uppercase }}</p>

currency，是什么过滤器——格式化数字为货币格式，单位是$符。

# 九、移动APP开发

## 1、移动端最小触控区域是多大？

移动端的点击事件的有延迟，时间是多久，为什么会有？ 怎么解决这个延时？（click 有 300ms 延迟,为了实现safari的双击事件的设计，浏览器要知道你是不是要双击操作。）

# 十、NodeJs

## 1、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。

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

至少给出自己的思路（url-hash,可以使用已有的一些框架history.js等）

## 3、Node.js的适用场景？

1)、实时应用：如在线聊天，实时通知推送等等（如socket.io）

2)、分布式应用：通过高效的并行I/O使用已有的数据

3)、工具类应用：海量的工具，小到前端压缩部署（如grunt），大到桌面图形界面应用程序

4)、游戏类应用：游戏领域对实时和并发有很高的要求（如网易的pomelo框架）

5)、利用稳定接口提升Web渲染能力

6)、前后端编程语言环境统一：前端开发人员可以非常快速地切入到服务器端的开发（如著名的纯Javascript全栈式MEAN架构）

## 4、(如果会用node)知道route, middleware, cluster, nodemon, pm2, server-side rendering么?

Nodejs相关概念的理解程度

## 5、解释一下 Backbone 的 MVC 实现方式？

流行的MVC架构模式

## 6、什么是“前端路由”?什么时候适合使用“前端路由”? “前端路由”有哪些优点和缺点?

熟悉前后端通信相关知识

前端路由就是在不进行后端请求的情况下对页面进行跳转

## 7、对Node的优点和缺点提出了自己的看法？

优点：

1. 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。

2. 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：

1. 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。

2. 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（第三方库现在已经很丰富了，所以这个缺点可以说不存在了）。

# 十一、前端概括性问题

## 1、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？

使用率较高的框架有jQuery、YUI、Prototype、Dojo、Ext.js、Mootools等。尤其是jQuery，超过91%。

轻量级框架有Modernizr、underscore.js、backbone.js、Raphael.js等。（理解这些框架的功能、性能、设计原理）

前端开发工具：Sublime Text 、Eclipse、Notepad、Firebug、HttpWatch、Yslow。

开发过的插件：城市选择插件，汽车型号选择插件、幻灯片插件。弹出层。（写过开源程序，加载器，js引擎更好）

## 对BFC规范的理解？

Formatting Context：指页面中的一个渲染区域，并且拥有一套渲染规则，他决定了其子元素如何定位，以及与其他元素的相互关系和作用。

## 3、99%的网站都需要被重构是那本书上写的？

网站重构：应用web标准进行设计（第2版）

## 4、WEB应用从服务器主动推送Data到客户端有那些方式？

html5 websoket

WebSocket通过Flash

XHR长时间连接

XHR Multipart Streaming

不可见的Iframe

<script>标签的长时间连接(可跨域)

## 5、加班的看法

加班就像借钱，原则应当是------救急不救穷

## 6、平时如何管理你的项目，如何设计突发大规模并发架构？

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 命民以该JS 功能为准英文翻译；

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

## 7、那些操作会造成内存泄漏？

内存泄漏指任何对象在您不再拥有或需要它之后仍然存在。

垃圾回收器定期扫描对象，并计算引用了每个对象的其他对象的数量。如果一个对象的引用数量为 0（没有其他对象引用过该对象），或对该对象的惟一引用是循环的，那么该对象的内存即可回收。

setTimeout 的第一个参数使用字符串而非函数的话，会引发内存泄漏。

闭包、控制台日志、循环（在两个对象彼此引用且彼此保留时，就会产生一个循环）

## 8、你说你热爱前端，那么应该WEB行业的发展很关注吧？ 说说最近最流行的一些东西吧？

Node.js、Mongodb、npm、MVVM、MEAN、react、angularjs、browserify、webpack

## 9、你有了解我们公司吗？说说你的认识？

因为我想去阿里，所以我针对阿里的说

最羡慕就是在双十一购物节，350.19亿元，每分钟支付79万笔。海量数据，居然无一漏单、无一故障。太厉害了。

携程、去哪儿：一个主要从事在线旅游O2O，当然也可以订酒店，买车票，买机票

## 10、移动端（比如：Android IOS）怎么做好用户体验?

融入自己的设计理念，注重用户体验，选择合适的技术

## 你所知道的页面性能优化方法有那些？

压缩、合并，减少请求，代码层析优化。。。

## 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？

知识面宽度，最好熟悉一些后台语言，比如php，展现出自己的技术两点

## 13、AMD（Modules/Asynchronous-Definition）、CMD（Common Module Definition）规范区别？

## 14、谈谈你认为怎样做能使项目做的更好？

考虑问题的深入，不仅仅停留在完成任务上，要精益求精

## 15、你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？

表现出对前端的认同与兴趣，关注相关技术前沿

## 16、php中下面哪个函数可以打开一个文件，以对文件进行读和写操作？

A.fget(); B.file\_open(); **C.fopen();** D.open\_file();

## 17、php中rmdir可以直接删除文件夹吗？该目录必须是空的，而且要有相应的权限--来自api

A.任何文件夹都可以删除 B.空文件夹可以删除

C.有权限的任何文件夹都可以删除 D.有权限的空文件夹可以删除

## 18、phpinset和empty的区别，举例说明

1、empty函数

用途：检测变量是否为空

判断：如果 var 是非空或非零的值，则 empty() 返回 FALSE。换句话说，""、0、"0"、NULL、FALSE、array()、var $var; 以及没有任何属性的对象都将被认为是空的，如果 var 为空，则返回 TRUE。注意：empty() 只检测变量，检测任何非变量的东西都将导致解析错误。换句话说，后边的语句将不会起作用;