框架类面试题

1. Angular.js
2. **MVC**

MVC指的是 Model View Controller 模型-视图-控制器设计模式，在前端中，我们可以简单的理解为：

V-View，视图层，一般是我们的html文件层，用于展示数据内容。

C-Controller，控制器，控制器帮助将M层数据给V,或者当View层数据有所改变时，通知M层，M层数据也做相应的改变。

M-Model，模型，当我们需要使用到数据的时候，数据暂存在这里。如var str="hello",或者从数据库中取出数据之后，将数据赋给一个对象或者变量，都可以理解为数据模型。

1. **mvvm 框架**

MVVM模式主要分为三部分：

View：它专注于界面的显示和渲染。

ViewModel：它是View和Model的粘合体，负责View和Model的交互和协作，它负责给View提供显示的数据，以及提供了View中Command事件操作Model的途径

Model：它是与应用程序的业务逻辑相关的数据的封装载体，它是业务领域的对象，Model并不关心会被如何显示或操作，所以模型也不会包含任何界面显示相关的逻辑。在web页面中，大部分Model都是来自Ajax的服务端返回数据或者是全局的配置对象

优点:

低耦合：View可以独立于Model变化和修改，同一个ViewModel可以被多个View复用；并且可以做到View和Model的变化互不影响

可重用性：可以把一些视图的逻辑放在ViewModel，让多个View复用

独立开发：开发人员可以专注与业务逻辑和数据的开发（ViewModemvvmdi计人员可以专注于UI(View)的设计

可测试性：清晰的View分层，使得针对表现层业务逻辑的测试更容易，更简单

1. **angular 的数据绑定采用什么机制？详述原理**

1）每个双向绑定的元素都有一个watcher

2）在某些事件发生的时候，调用digest脏数据检测。

这些事件有：表单元素内容变化、Ajax请求响应、点击按钮执行的函数等。

3）脏数据检测会检测rootscope下所有被watcher的元素。

$digest函数就是脏数据监测

1. **什么是脏检查？**

将原对象复制一份快照，在某个时间，比较现在对象与快照的值，如果不一样就表明发生了变化，这个策略要保留两份变量，而且要遍历对象，比较每个属性，这样会有一定的性能问题

1. **angular 核心？**

AngularJS是为了克服HTML在构建应用上的不足而设计的。 AngularJS有着诸多特性，最为核心的是：

MVC

模块化

自动化双向数据绑定

语义化标签、依赖注入等等

1. **解释下什么是$rootScrope以及和$scope的区别？**

通俗的说$rootScrope 页面所有$scope的父亲。

1. **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。

1. **Angular中的digest周期是什么？**

每个digest周期中，angular总会对比scope上model的值，一般digest周期都是自动触发的，我们也可以使用$apply进行手动触发。更深层次的研究，可以移步The Digest Loop and apply。

1. **Angular Directive中restrict 中分别可以怎样设置？scope中@,=,&有什么区别？**

restrict中可以分别设置:

A匹配属性

E匹配标签

C匹配class

M 匹配注释

当然你可以设置多个值比如AEC,进行多个匹配。

在scope中，@,=,&在进行值绑定时分别表示

@获取一个设置的字符串，它可以自己设置的也可以使用{{yourModel}}进行绑定的;

= 双向绑定，绑定scope上的一些属性；

& 用于执行父级scope上的一些表达式，常见我们设置一些需要执行的函数

1. **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

1. **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

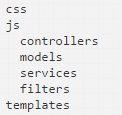
Service：通过构造函数方式创建 service，返回一个实例化对象

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

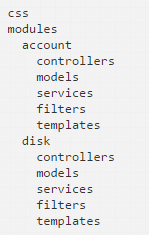
1. **一个 angular 应用应当如何良好地分层？**

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：



但是对于规模较大的项目，最好按业务模块划分，比如



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

1. **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

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。

1. **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，效率会高一些，

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。

1. **优化angular应用性能的方法**

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）

1. **详述 angular 的 “依赖注入”**

依赖注入是一种软件设计模式，用来处理代码的依赖关系，降低耦合度，通俗的讲就是一个类需要使用另一个类，不用再自己去new，而是由提供者主动提供

依赖注入可以分为三个角色：调用者，注入器（Injector）和服务者（所依赖的对象）

在angular中我用到最多的就是组件中注入服务，也就是创建一个独立的服务，然后再app.module中声明我们的服务，最后在组件的构造函数中（只能在构造函数中）声明注入器，这样就可以将我们的服务注入到该组件

这样我们可以实现可重用的组件

1. **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。

1. **angularjs实现网络请求**

$http

1. **ng-if和ng-show的区别**

ng-if是根据值判断是否添加某个元素，而ng-show是改变元素的display属性决定元素的显示与否

1. **如何自定义指令**

app.directive(指令名,配置对象)

1. **如何使用路由**

首先引入路由模块的js文件

 -设置ng-view

 -设置路由跳转链接

-配置路由信息

 $routeProvider.when(路径,模板信息);

1. **Angular4相关**

相对于1版本而言，4版本删除了ng-controller、ng-show、ng-hide指令，去除了$scope；优化了路由嵌套、过滤器、依赖注入机制等，增强了对移动端的支持，angular4的核心是组件化，主要使用的是typescript，与2版本一致

1. **组件的生命周期**

constructor：构造器函数，一般用于注入服务

ngOnChanges：检测到输入数据变化，首次触发发生在ngOnInit前。

ngOnInit：组件初始化，通常会设置一些初始值

ngDoCheck：手动触发更新检查

ngOnDestroy：组件注销时的清理工作

1. **管道**

Angular4中的管道就相当于之前版本的过滤器，主要是用于格式化数据源，通过导入pipe，在pipe的实现类中的transform方法中实现相关的过滤效果。

1. **路由**

路由主要是为了实现单页面应用界面的跳转，在ng中，通过routermodule实现一个路由模块，通过routes配置路由信息，包括路径、对应的组件，及子路由等，然后再需要的地方设置路由出口<router-outlet>，通过routerlink进行页面的跳转

1. Vue.js
2. **简单介绍下vue.js**

Vue.js是JavaScript MVVM（Model-View-ViewModel）库，十分简洁，Vue核心只关注视图层，相对AngularJS提供更加简洁、易于理解的API。Vue尽可能通过简单的API实现响应的数据绑定和组合的视图组件。

1. **Vue.js特点**

简洁：页面由HTML模板+Json数据+Vue实例组成

数据驱动：自动计算属性和追踪依赖的模板表达式

组件化：用可复用、解耦的组件来构造页面

轻量：代码量小，不依赖其他库

快速：精确有效批量DOM更新

模板友好：可通过npm，bower等多种方式安装，很容易融入

1. **什么是组件，组件的好处是什么，如何实现组件**

**组件可以理解为一个封装好的UI模块，就类似于一个html标记；组件化可以扩展HTML元素，封装可重用的代码。提高代码的复用性。**

**在vue中通过Vue.component()函数创建一个全局组件，通过对象直接量方式创建局部组件**

**组件中的data必须是一个函数，函数中返回包含了数据的对象**

1. **Vue实例的生命周期**

Mounted（绑定）——updated（更新）——destroyed（销毁）

1. **组件的生命周期**

beforeCreate、created、mounted、updated、destroyed

1. **Vuex**

通俗来讲，Vuex 就是前端为了方便数据的操作而建立的一个数据存储容器，vuex解决了组件之间共享同一状态的麻烦问题。当我们的应用遇到多个组件共享状态时，会需要；

vuex 就是使用一个 store 对象来包含所有的应用层级状态，一个 store 有四个属性：state， getters, mutations, actions

state 上存放的，说的简单一些就是变量，也就是所谓的状态

getters上简单来说就是存放一些公共函数供组件调用

mutations 上存放的一般就是我们要改变 state 的一些方法。

Action 类似于 mutation，不同在于：Action 提交的是 mutation，而不是直接变更状态，它管理触发条件的

1. **axios**

vue2之后，不再继续维护vue-resource，并推荐大家使用 axios，其主要作用是处理数据请求，用法简单，可以直接使用axios.get、axios.post等发送请求

1. React.js
2. React.js是什么框架

React.js是一个view层的框架，主要推崇的是组件化，不擅长处理数据，所以也不能代替类似MVC、MVVM的框架

1. React组件中属性的监听

通过组件的props的属性去访问一个属性，比如props.name，但是props.children是获取所有的子节点

1. React中如何获取真实的DOM节点

通过给元素添加ref属性，并在相关函数中使用this.refs去获取

1. 如何实现用户的交互，检测数据的变化

通过this.state属性去处理，当数据变化时，可以用个this.setState()函数去设置，然后通过this.state.变量名去获取

1. 组件的生命周期

-Mounting：已经插入真实dom

-Updating：正在被重新渲染

-Unmounting：已移除真实的DOM

1. React中如何实现接口请求

Ajax