面试题

1，说说对es6的理解（说一下es6，知道es6吗）

语法糖(箭头函数，类的定义，继承)，以及一些新的扩展（数组，字符串，对象，方法等），对作用域的重新定义，以及异步编程的解决方案（promise，async，await）、

2，说说你对Promise的理解

Promise 是异步编程的一种解决方案，比传统的解决方案——回调函数和事件监听——更合理和更强大。Promise 有三种状态：pending（进行中）、fulfilled（已成功）和rejected（已失败）。但是无法获取到pending状态，在promise中接受两个内置参数分别是resolve（成功）和reject（失败），Promise实例生成以后，可以用then方法分别指定resolved状态和rejected状态的回调函数。then方法可以传递两个回调函数第一个是成功，第二个是失败，失败回调也可以使用promise的catch方法回调，promise还有一个强大的功能那就是all方法可以组合多个promise实例，包装成一个新的 Promise 实例。

3， angular双向数据绑定原理

所谓的双向绑定，其实就是从界面的操作能实时反映到数据，数据的变更能实时展现到界面。AngularJS在$scope变量中使用脏值检查来实现了数据双向绑定，并且可以通过$scope.$watch来监听变化触发回调；

angular中使用的是脏检查机制，在angular中每次你绑定一些东西到你的UI上时你就会往$watch队列里插入一条$watch，当我们的模版加载完毕时，也就是在linking阶段（Angular分为compile阶段和linking阶段---译者注），Angular解释器会寻找每个directive，然后生成每个需要的$watch。当浏览器接受到可以被angular context处理的事件时就会触发digest循环，这个循环是由两个更小的循环组合起来的，一个是$watch列表，一个是$evalAsync列表，而$watch列表在$digest循环中被“脏值检查”解析，在digest将会遍历我们的watch，然后询问它是否有属性和值的变化，直到$watch队列都检查过，在检查数据变化的时候，由于并不知道这个事件是对哪些数据进行了更改，以及这个事件有可能造成事件之外的其他任何地方的数据更改，所以必须进行一次大检查，将所有“注册”过的值全部检查一遍，一次检查称为一个周期，每次最少检查两遍，因为第二遍用来确认，前一遍的变动中是否有数据的变动，导致了其他数据的变动，如果第二次有变动的话，会再执行一遍，直到最后两次完全一致，则停止检查（其实就是个（递归（遍历））的过程），考虑到内存的消耗和死循环的风险，脏检查每个周期最多递归执行10遍，如果超过10遍就会抛出一个错误。当$digest循环结束时，DOM相应地变化。

在angular中

ng-click，ng-change，ng-blur...就是对各类用户事件的封装

$timeout，$http，$window，$location...就是对各种JS/API事件的封装

ng-model，以及控制器中的数据，就是对值的“注册”

$scope 本质是一个总的事件逻辑的封装容器，同时抽象为数据载体，实质上数据都存在于浏览器堆内存中

$scope.apply() & $scope.digest() 即Angular中的“数据大检查”的function

所以如果我们使用了非Angular封装的事件改编数据时，要手动执行一次大检查

由于Angular这种脏检查的方法效率不高，如果一个页面绑定的view超过2000个，就可能存在比较明显的性能问题，官方称之为“脏检查”

4，vue实现数据双向绑定的原理

vue实现数据双向绑定主要是采用数据劫持结合发布者-订阅者模式的方式，通过Object.defineProperty()来劫持各个属性的setter，getter，在数据变动时发布消息给订阅者，触发相应的监听回调。具体话其实就是通过Obeject.defineProperty() 中的setter和getter来监听属性变动实现Observer进行数据的监听然后就是通知订阅者，那么订阅者其实就是简单的一个数组，这个数组中的内容就是我门使用了的一个数据的集合，使用了的数据可以通过getter得到，其实就是在调用的时候给数组里面添加一个订阅者这样就是实现了一个Watcher（需要监听的数据的集合），然后在实现一个Compile其作用就是解析模板指令，将模板中的变量替换成数据，然后初始化渲染页面视图，并将每个指令对应的节点绑定更新函数，添加监听数据的订阅者，一旦数据有变动，收到通知，更新视图，其实vue的数据双向绑定就是MVVM作为数据绑定的入口，整合Observer、Compile和Watcher三者，通过Observer来监听自己的model数据变化，通过Compile来解析编译模板指令，最终利用Watcher搭起Observer和Compile之间的通信桥梁，达到数据变化 -> 视图更新；视图交互变化(input) -> 数据model变更的双向绑定效果。

5，vue路由的实现原理

在vue中路由主要有 hash与History interface两种方式实现前端路由，单页路由的特点就是采用前端路由系统，通过改变URL，在不重新请求页面的情况下，更新页面视图。目前在浏览器环境中这一功能的实现主要有两种方式：hash和History interface，这两种模式的实现分别是

先说hash，在浏览器的url中hash（“#”）符号的本来作用是加在URL中指示网页中的位置：#符号本身以及它后面的字符称之为hash，可通过window.location.hash属性读取。它具几个特点就是：hash虽然出现在URL中，但不会被包括在HTTP请求中。它是用来指导浏览器动作的，对服务器端完全无用，因此，改变hash不会重新加载页面，并且可以为hash的改变添加hashchange监听事件，在一个就是每一次改变hash（window.location.hash），都会在浏览器的访问历史中增加一个记录，我就可以通过他的这几个特点实现一个hash模式的单页路由，通过对location.hash的修改实现push方法（跳转页面），通过对location.href的修改实现replace()方法，通过对hashchange事件的监听实现页面跳转后的数据更新

而History模式则是完全采用了h5的新特性，从HTML5开始，History interface提供了两个新的方法：pushState(), replaceState()使得我们可以对浏览器历史记录栈进行修改，以及popState事件可以监听到状态的变更

不过history模式有一个问题就是

对于单页应用来讲，理想的使用场景是仅在进入应用时加载index.html，后续在的网络操作通过Ajax完成，不会根据URL重新请求页面，但是如果用户直接在地址栏中输入并回车，浏览器重启重新加载的时候history模式则会将URL修改得就和正常请求后端的URL一样，在此情况下重新向后端发送请求，如后端没有配置对应 的路由处理，则会返回404错误。这种问题的解决，一般情况下我们都是在后端进行配置，将所有的路由请求都指向index.html文件

6，Cookie和localstorage、sessionstorage的区别

Cookie技术浏览器兼容性好，但操作比较复杂，需要程序员自己封装，源生的Cookie接口不友好, 存储的内容较小， cookie的数据会随着ajax的请求发送到服务端，一般情况主要用在用户登录的时候我们可以通过在 Cookie 中存入一段辨别用户身份的数据，用于后台判断。

WebStorage则不能超过8MB，操作简单；可以代替一些cookie的工作，一般主要是用于存储一些本地数据，购物车数据之类的

在安全方面的话，都不安全，一般就是对数据进行一些简单的加密，如base64编码，加密约定之类的东西

localstorage、sessionstorage一个是长期存储，一个是会话存储

7，fetch与ajax的区别？

Ajax主要是利用的是XMLHttpRequest对象来请求数据的。

Fetch是window的一个方法 主要特点是

1、第一个参数是URL

2、第二个参数可选参数 可以控制不同的init对象

3、使用了js 中的promise对象

fetch 的第二参数中

1、默认的请求为get请求 可以使用method:post 来进行配置

2、第一步中的 response有许多方法 json() text() formData()

3、Fetch跨域的时候默认不会带cookie ，如果你想在fetch请求里附带cookies之类的凭证信息，可以将 credentials参数设置成 “include” 值。

还有就是

fetch返回的promise在某些错误的http状态下如400、500等不会reject的错误状态，相反它会被resolve；只有网络错误会导致请求不能完成时，fetch 才会被 reject；所以一般会对fetch请求做一层封装，在resolve中真对于大于200和 小于300的状态返回正确信息，其他则返回错误信息

所有的IE浏览器都不会支持 fetch()方法

左边200像素，右边自适应如何实现？

Position有哪些值？作用

3、

4、一像素边框怎么实现？

5、移动端怎么自适应？

7、如何开启gpu加速

8、页面如何优化

9、你们用什么跨域？原理是什么

10、闭包？用在哪些地方？

11、水平垂直居中

12、你们跟后台是怎么交流的

13、项目流程

14、项目中遇到过什么bug

15、Html5新标签

16、Svg和canvas的区别

17、实现一个loading动画

18、登陆怎么做的

19、Cookie和localstorage、sessionstorage的区别

20、Axios的配置

21、数组的map、filter、forEach

22、浏览器的兼容，移动端适配

23、Get、post的区别

1、fetch与ajax的区别？

2、怎么上传文件？

3.用node发过包吗，步骤是什么

4.vue做路由怎么做，具体配置，怎么传参，对vuex的理解，具体实现？

5.webpack打包的配置，常见的loaders和plugins？

6.vue的脚手架，有个Server

7.express的中间有哪些？路由怎么做？

1.webpack2的常用的loaders，js的插件，plugins

2.版本的管理工具，是用通讯化工具，还是直接敲的命令行

3.@click.native，在组件上触发点击时间

4.vuex用的多吗，它的构成

5.grunt和gulp？

6.对象深复制？

7.用计算属性对所有商品的价格或者ID排序 vue1用orderby

8.前后端分离，API谁写的

9.用过lazyload吗？是干什么的？

1.你们怎么嵌套h5页面的，

3.处理过浏览器的兼容问题吗？

4.webpack和gulp项目里你是怎么用的，

5.onload和ready的区别

6.jquery为什么可以支持链式调用，原理是什么？

7.webapp你主要做的是哪个模块，是页面布局做的多还是交互做的多

8.call和apply是干什么用的，有什么区别，平时都是怎么用的。

9.用到过哪些es6

10.let和var的区别

11.自己写过jq的插件吗，写过哪些插件

12.bootstrap用吗，响应式布局的原理

15.说一下计算属性

16.vue的兄弟组件传递

17.路由怎么实现页面跳转

18.vue传参的方法有哪些

19.路由跳转之后，比如从a页面跳到b页面，a页面会不会销毁，怎么把b页面的值带到a页面，怎么让a页面不销毁（这是一种很常见的需求），组件销毁的标志，调用distory方法

20.localstoge个setionstorage的区别

21.用过pc的模板吗，

22.说一下js模板引擎。

23.react中state与props的区别?

24.es6如何转为es5？

25.说一下原型链、原型链继承。