#### 【已面试题目:】

1. 列举HTML5 的标签：

基础：<html> <body> <title> <h1>to<h6> <p> <br> <hr>

格式：<b> <abbr> <address> <center> <del> ......

表单：<form> <input> <textarea> <button> <label> ......

框架：<iframe>

图像：<img> <map> <area> <canvas> <figcaption> <figure>

Audio/Video：<audio> <source> <track> <video>

链接：<a> <link> <nav>

列表：<ul> <ol> <li> ......

表格：<table> <th> <tr> <td> ......

样式/节：<style> <div> <span> ......

元信息：<head> <meta> <base>

程序：<script> <noscript> ......

1. jquery 中的循环：

（1）for 循环：

var $input = $( 'input' );

for( var i=0; i < $inputs.length; i++ )

（2）for ... of 循环：

var $input = $( 'input' );

for( var input of $inputs )

遍历方法：each（）方法：

$("li").each ( function ( ) {

alert( $( this ). text( ) )

});

1. jquery 中的 $(this) 和 JavaScript 中的 this 区别：

（1）$(this)是 jquery 对象，能调用 jquery 的方法，例如 click(), keyup(), ....

（2）this永远指向函数运行时所在的对象！而不是函数被创建时所在的对象。

this对象是在运行时基于函数的执行环境绑定的，在全局环境中，this等于window

1. vue 中路由传参的方式： 2种
2. params：

只能使用name，不能使用path；

参数不会显示在路径上；

浏览器强制刷新参数会被清空。

// 传递参数

this.$router.push({

name: Home，

params: {

number: 1 ,

code: '999'

}

})

// 接收参数

const p = this.$route.params

（2）query：

参数会显示在路径上，刷新不会被清空；

name 可以使用path路径。

// 传递参数

this.$router.push({

name: Home，

query: {

number: 1 ,

code: '999'

}

})

// 接收参数

const q = this.$route.query

1. typeof 返回的数据类型：6种

object undefined string number boolean function

1. vue 中路由导航守卫有哪几种：

（1）全局守卫：

router.beforeEach((to, from, next) => { })

router.afterEach(( to, from ) => { })

（2）组件内的守卫：

beforeRouteEnter：( to, from, next ) => { }

beforeRouteEnter：( to, from, next ) => { }

beforeRouteLeave：( to, from, next ) => { }

（3）路由独享的守卫：( 定义在router路由数组的路由对象中 )

beforeEnter：( to, from, next ) => { }

1. ajax跨域请求问题的五种解决方案：
2. 使用跨域资源共享代理(corsproxy)

方式一: 安装node.js客户端

方式二: 在要访问的真实路径前加上www.corsproxy.com，即通过代理的方式来实现跨域访问, 但是不适合高并发和高可用的应用情景

1. 使用jsonp

原理: ajax不能跨域请求但是JavaScript可以跨域访问, 因此可以把要返回的json格式的数据包装到js函数中, 然后在页面中定义该函数即可显示

实现: 在ajax的请求中将dataType定义为"jsonp"

1. 使用XmlHttpRequest

原理: 其实XmlHttpRequest是ajax的底层实现

实现:

首先，新建一个XMLHttpRequest的实例。

var xhr = new XMLHttpRequest();

然后，向远程主机发出一个HTTP请求。

xhr.open('GET', 'example.php');

xhr.send();

接着，就等待远程主机做出回应。这时需要监控XMLHttpRequest对象的状态变化，指定回调函数。

xhr.onreadystatechange = function(){

if ( xhr.readyState == 4 && xhr.status == 200 ) {

alert( xhr.responseText );

} else {

alert( xhr.statusText );

}

};

1. 使用httpclient

httpclient是一个可以接收和发送http请求的工具包, 里面的doGet和doPost可以满足我们常见的需求, 同时还可以选择带参数或者不带参数

1. 使用阿里巴巴的Dubbo分布式服务框架

dubbo是分布式服务框架, 阿里巴巴soa服务化治理方案的核心框架(现已停止更新)

该框架采用生产者--消费者模型

执行流程:

1、在服务容器加载的时候, 运行服务的生产者

2、服务生产者在启动的时候，向简易注册中心注册自己能提供的服务

3、消费者在启动的时候，向注册中心订阅自己需要的服务

4、注册中心返回服务提供者地址列表给消费者，如果有变更，注册中心将基于长连接推送变更数据给消费者

5、服务消费者，从提供者地址列表中，基于软负载均衡算法，选一台提供者进行调用，如果调用失败，再选另一台调用

6、服务消费者和提供者，在内存中累计调用次数和调用时间，定时每分钟发送一次统计数据到监控中心

备注：

简易注册中心一般使用zookeeper（当然，这里使用有点大材小用）

1. storage和cookie ：

相同之处：

1. 它们都可以用于存储用户数据
2. 它们存储数据的格式都是字符串形式

（3）它们存储的数据都有大小限制

不同之处：

1. 它们的生命周期不同。sessionStorage 的生命周期是一个会话，localStorage的生命周期是永久，cookie 的生命周期可以自定义，cookie 可以设置过期时间，数据在过期时间之前可以访问。
2. 它们的存储大小限制不同。大部分现代浏览器 Storage 的存储限制大小为 5M，cookie 的存储大小限制 为 4K。

（3）浏览器支持不同，API 调用方式不同。

【localStorage和sessionStorage区别：】

1. localStorage和sessionStorage一样都是用来存储客户端临时信息的对象。他们均只能存储字符串类型的对象（虽然规范中可以存储其他原生类型的对象，但是目前为止没有浏览器对其进行实现）。
2. localStorage生命周期是永久，这意味着除非用户显示在浏览器提供的UI上清除localStorage信息，否则这些信息将永远存在。

（3）sessionStorage生命周期为当前窗口或标签页，一旦窗口或标签页被永久关闭了，那么所有通过sessionStorage存储的数据也就被清空了。

（4）不同浏览器无法共享localStorage或sessionStorage中的信息。相同浏览器的不同页面间可以共享相同的 localStorage（页面属于相同域名和端口），但是不同页面或标签页间无法共享sessionStorage的信息。这里需要注意的是，页面及标 签页仅指顶级窗口，如果一个标签页包含多个iframe标签且他们属于同源页面，那么他们之间是可以共享sessionStorage的。

1. <keep-alive></keep-alive>的作用是什么：

<keep-alive> 是Vue的内置组件，能在组件切换过程中将状态保留在内存中，防止重复渲染DOM。

【<keep-alive>的生命周期:】

当引入keep-alive 的时候，页面第一次进入，钩子的触发顺序created-> mounted-> activated，退出时触发deactivated。当再次进入（前进或者后退）时，只触发activated。

1. vue-router中mode的hash和history的区别：
2. hash 模式（默认）下，仅 hash 符号（即地址栏 URL 中的 # 符号，比如这个 URL：http://www.abc.com/#/hello，hash 的值为 #/hello）之前的内容会被包含在请求中，如 http://www.abc.com，因此对于后端来说，即使没有做到对路由的全覆盖，也不会返回 404 错误。
3. history 模式下，前端的 URL 必须和实际向后端发起请求的 URL 一致，如 http://www.abc.com/book/id。如果后端缺少对 /book/id 的路由处理，将返回 404 错误。
4. Vue单页面应用的优缺点：

优点：（1）用户体验好，快，内容的改变不需要重新加载整个页面，基于这一点spa对服务器压力较小。

（2）前后端分离。

（3）页面效果会比较炫酷（比如切换页面内容时的专场动画）。

缺点：（1）不利于seo。

（2）导航不可用，如果一定要导航需要自行实现前进、后退。（由于是单页面不能用浏览器的前进后退功能，所以需要自己建立堆栈管理）。

（3）初次加载时耗时多。

（4）页面复杂度提高很多。

1. vue按需加载：路由懒加载 和 组件懒加载
2. 路由懒加载

【第一种方法：】vue异步组件实现懒加载

　component：resolve => (require(['需要加载的路由的地址'])，resolve)

【第二种方法；】ES提出的import方法（最常用）

const Home = ( ) => import('需要加载的模块地址')

component: Home

1. 组件懒加载

【const方法：】

<template>

<div class="hello">

<One-com></One-com>

1111

</div>

</template>

<script>

const One = ()=>import("./one");

export default {

components:{

"One-com":One

},

data () {

return {

msg: 'Welcome to Your Vue.js App'

}

}

}

</script>

【异步方法：】

<template>

<div class="hello">

<One-com></One-com>

1111

</div>

</template>

<script>

export default {

components:{

"One-com":resolve=>(['./one'],resolve)

},

data () {

return {

msg: 'Welcome to Your Vue.js App'

}

}

}</script>

#### 【收藏高频面试题】

1、Vue的双向数据绑定原理是什么？

vue.js 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。

具体步骤：

第一步：需要observe的数据对象进行递归遍历，包括子属性对象的属性，都加上 setter和getter

这样的话，给这个对象的某个值赋值，就会触发setter，那么就能监听到了数据变化

第二步：compile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

第三步：Watcher订阅者是Observer和Compile之间通信的桥梁，主要做的事情是:

1、在自身实例化时往属性订阅器(dep)里面添加自己

2、自身必须有一个update()方法

3、待属性变动dep.notice()通知时，能调用自身的update()方法，并触发Compile中绑定的回调，则功成身退。

第四步：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。

（ps：此答案同样适合”vue data是怎么实现的？”）

2、请说下封装 vue 组件的过程？

首先，组件可以提升整个项目的开发效率。能够把页面抽象成多个相对独立的模块，解决了我们传统项目开发：效率低、难维护、复用性等问题。

然后，使用Vue.extend方法创建一个组件，然后使用Vue.component方法注册组件。子组件需要数据，可以在props中接受定义。而子组件修改好数据后，想把数据传递给父组件。可以采用emit方法。

3、请简述web开发中的兼容问题

（1）浏览器默认的内外边距不一样，所以用通用选择器\*设置margin和padding来设置。

（2）块标签设置浮动后，有设置margin的情况下，在ie6下的margin比别的浏览器大。

（3）img标签会有默认的间距，需要用浮动去设置

（4）火狐浏览器中的点击事件和滚动事件需要加（event）来兼容

（5）div里的内容，ie默认为居中，firefox默认为左对齐，需要用margin:0px auto来调节

（6）css3的动画效果不兼容ie8以下。需要使用js去写动画。

（7）ie6的双倍边距，比如设置了margin:10px,ie6中默认为20px，需要使用display:inline-block。