# **赞同科技**

1. data为什么是函数?

在 new Vue() 中， data 是可以作为一个对象进行操作的，然而在component 中， data 只能以函数的形式存在，不能直接将对象赋值给它。

为什么：

// 1.不是函数会报错

// 2.不是函数数据会被该组件的不同实例共享

(因为复用性 如果不是函数的话会污染 不能是对象 因为对象没有作用域)

1. jQuery和vue的区别?

jQuery是使用选择器（ $ ）选取DOM对象，对其进行赋值、取值、事件绑定等操作，其实和原生的HTML的区别只在于可以更方便的选取和操作DOM对象，而数据和界面是在一起的。比如需要获取label标签的内容：$("lable").val(); ,它还是依赖DOM元素的值。Vue则是通过Vue对象将数据和View完全分离开来了。对数据进行操作不再需要引用相应的DOM对象，可以说数据和View是分离的，他们通过Vue对象这个vm实现相互的绑定。这就是传说中的MVVM。

jQuery是操作DOM元素 vue是封装使用虚拟DOM

1. 单向,双向数据流的区别?

单向绑定：插值形式{undefined{data}}，v-bind也是单向绑定

双向绑定：v-model，用户对View层的更改会直接同步到Model层

1. MVVM是什么?

MVVM 是 Model-View-ViewModel 的简写。它本质上就是 MVC 的改进版。它本质上就是 MVC 的改进版。MVVM就是将其中的 View 的状态和行为抽象化，让我们将视图 UI 和业务逻辑分开。

1. 说一下NextTick

当你修改了data 的值然后马上获取这个 dom 元素的值，是不能获取到更新后的值， 你需要使用 $nextTick 这个回调，让修改后的 data 值渲染更新到 dom 元素之后在获取，才能成功。

1. vue的生命周期? 10个

加载时

beforeCreate : 开始创建

data 和 methods 都未创建, 此处不能使用

created : 创建完毕

data 和 methods 创建完毕, 最早的可以使用处

beforeMount : 开始挂载

内存中已编译好所有内容, 准备显示到页面

mounted : 挂载完毕

组件脱离创建阶段, 真正显示到页面上. 操作页面的DOM 最早可以在这里进行

更新

beforeUpdate : 更新前

updated : 更新完毕

keep-alive 相关

activated : 被 keep-alive 缓存的组件激活时调用。

deactivated : 被 keep-alive 缓存的组件停用时调用。

销毁

beforeDestory : 销毁前

清除定时器 回收不要的数据

destroyed : 销毁完毕

data 和 methods 此处已消失, 无法使用

1. 路由的钩子函数？？

　第一种：全局钩子函数。  
　　router.beforeEach

第二种:单个路由钩子函数  
　　 beforeEnter

第三种:组件级钩子函数  
　　beforeRouteEnter

1. 怎么扩展组件?

常见的组件扩展方法有：mixins,slots,extends等。

1. Promise的几个方法?

[reject的用法](https://blog.csdn.net/qq_34645412/article/details/81170576" \l "t1" \t "https://blog.csdn.net/qq_34645412/article/details/_self)

[catch的用法](https://blog.csdn.net/qq_34645412/article/details/81170576" \l "t2" \t "https://blog.csdn.net/qq_34645412/article/details/_self)

[all的用法](https://blog.csdn.net/qq_34645412/article/details/81170576" \l "t3" \t "https://blog.csdn.net/qq_34645412/article/details/_self)

[race的用法](https://blog.csdn.net/qq_34645412/article/details/81170576" \l "t4" \t "https://blog.csdn.net/qq_34645412/article/details/_self)

1. Beforedestroy用来干什么的?

清除定时器 回收不要的数据

# **中软国际**

1. 块级格式化上下文应用?(BFC)

如清除浮动，自适应布局等。

- 是一块区域body 根元素

- 浮动元素：float 除 none 以外的值

- 绝对定位元素：position (absolute、fixed)

- display 为 inline-block、table-cells、flex

- overflow 除了 visible 以外的值 (hidden、auto、scroll)

1. 函数的节流和防抖?

防抖=>防抖是在事件被触发n秒后再执行回调函数，如果在这n秒内又被触发，则重新计时。

节流=>简单粗暴的讲，就是在我不断地触发事件的时候别老是发请求，发一次就够了。（高频事件触发，一定时间内只要执行一次，节流可以减少不断执行频率。)

1. Js new一个对象会发生什么?

- 创建一个空对象,将它的引用赋给 this,继承函数的原型。

- 通过 this 将属性和方法添加至这个对象

- 最后返回 this 指向的新对象,也就是实例(如果没有手动返回其他的对象)

1. 对原型链的了解

- 对象和构造函数之间连接的一个链条

- 原型链的作用 可以通过原型链继承构造函数对象定义的一些公共的属性方法

1. JS的深拷贝浅拷贝

- 假设B复制了A，当修改A时，看B是否会发生变化，如果B也跟着变了，说明这是浅拷贝，如果B没变，那就是深拷贝

1. 如何解决页面出现卡顿,从哪些方面解决问题?



# **博为峰**

1. 什么是跨域

解决同源策略 代理模式 proxy

1. Vuex五个基础原理

state => 基本数据(data)

getters => 从基本数据派生的数据 ，和组件中的computed很相似

mutations => 提交更改数据的方法，同步！

actions => 像一个装饰器，包裹mutations，使之可以异步。

modules => 模块化Vuex

mutations是同步操作 actions是异步操作

1. 弹性布局

display:flex

flex-direction 容器内元素的排列方向(默认横向排列)

flex-wrap 容器内元素的换行(默认不换行)

justify-content 元素在主轴（页面）上的排列

align-items 元素在主轴（页面）当前行的横轴（纵轴）方向上的对齐方式

align-content 在弹性容器内的元素没有占用交叉轴上所有可用的空间时对齐容器内的各项（垂直）

1. Vuex的状态刷新,会不会保存记录

Vuex不会 想保存可以用到cookie 具体看下边连接

<https://www.cnblogs.com/xieyao/p/13265947.html>

1. Axios基于什么实现

基于promise的http库

1. Promise的异步是什么

Async

# 

# **深圳拓保**

1. JS的数据类型

基础类型 Number、String 、Boolean、Null和Undefined

引用类型 也就是对象类型Object type，

1. 判断引用类型的方法及原理

只要在这个实例的原型链上的类,用instanceof检测的时候都为true

1. 什么是闭包

闭包就是能够读取其他函数内部变量的函数。

1. 如何清除浮动

给浮动的元素的祖先元素加上高度

overflow:hidden;的本意是将超出父类的部分隐藏

clear:both; clear就是清除,both指的是左浮动,右浮动都要清除.意思就是:清除别人对我的影响.(伪类)

1. 两边给固定宽高 中间自适应 怎么操作

<div class="parent">

<div class="left">左边盒子宽度固定</div>

<div class="center">中间盒子 flex: 1; 实现自适应</div>

<div class="right">右边盒子宽度固定</div>

</div>

1. Vue-router的两种模式及原理

历史模式和哈希模式

hash原理:hash通过监听浏览器的onhashchange()事件变化，查找对应的路由规则  
history原理:利用H5的 history中新增的两个API pushState() 和 replaceState() 和一个事件onpopstate监听URL变化

Hash模式不刷新页面所以不会有404情况,history模式刷新会有404情况,修改Nginx配置文件

1. 说一下nextTick

$nextTick是在下一次dom更新循环结束之后执行延迟回调，在修改数据之后使用这个方法，立即更新dom.

1. 事件修饰符

.stop 阻止事件冒泡

.prevent 阻止默认行为

.capture 事件捕获

.self 自身触发

.once 只触发一次

1. 原型和原型链

- 对象和构造函数之间连接的一个链条

- 原型链的作用 可以通过原型链继承构造函数对象定义的一些公共的属性方法

1. v-model的绑定原理

v-model本质上是:value和v-on的结合体,就是绑定他的value,通过v-on触发，从而更新数据

双向绑定得的实现主要依赖于Object.defineProperty(),通过这个函数可以监听到get,set事件

1. NextTick和setTimeOut谁先执行,为什么?

NextTick先执行 执行顺序：微任务 > 宏任务

34.如何使样式独立?为什么?

Scoped 自定义属性, 会给所有要修改的dom添加了一个属性选择器

# **中电金信**

35.v-if与v-show的区别

v-if:

通过删除DOM元素实现元素的隐藏

惰性: 只有条件为真时, 才会加载元素到DOM

v-show:

通过设置元素的css样式: display:none 实现元素的隐藏, 不操作DOM.

非惰性: 不管条件真与假, 都会加载元素到 DOM

所以

v-if 的开销比 v-show 更大

v-show 有更高的初始化渲染消耗

适用场景

一个元素频繁进行 隐藏 和 显示 操作, 使用 v-show 更加合适

一个元素不频繁进行 隐藏 和 显示操作, 使用 v-if 更合适.

例如: 需要网络请求 成功后才显示的内容(自己简化理解记)

36.Vue的生命周期

加载时

beforeCreate : 开始创建

data 和 methods 都未创建, 此处不能使用

created : 创建完毕

data 和 methods 创建完毕, 最早的可以使用处

beforeMount : 开始挂载

内存中已编译好所有内容, 准备显示到页面

mounted : 挂载完毕

组件脱离创建阶段, 真正显示到页面上. 操作页面的DOM 最早可以在这里进行

更新

beforeUpdate : 更新前

updated : 更新完毕

keep-alive 相关

activated : 被 keep-alive 缓存的组件激活时调用。

deactivated : 被 keep-alive 缓存的组件停用时调用。

销毁

beforeDestory : 销毁前

destroyed : 销毁完毕

data 和 methods 此处已消失, 无法使用

(理解记,简单记)

37.Vue的双向数据绑定

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。(官方答案仅供参考)

38.事件修饰符

.stop 阻止事件冒泡

.prevent 阻止默认行为

.capture 事件捕获

.self 自身触发

.once 只触发一次

39.组件间的通信(传参)

父子传参,子父传参,兄弟传参

40.说一下keep-alive

keep-alive是一个抽象组件：它自身不会渲染一个DOM元素，也不会出现在父组件链中；使用keep-alive包裹动态组件时，会缓存不活动的组件实例，而不是销毁它们。(看自己理解答案仅供参考)

41.Var let const的区别

（1）块级作用域： 块作用域由 { } 包括，let和const具有块级作用域，var不存在块级作用域。块级作用域解决了ES5中的两个问题：

内层变量可能覆盖外层变量

用来计数的循环变量泄露为全局变量

（2）变量提升： var存在变量提升，let和const不存在变量提升，即在变量只能在声明之后使用，否在会报错。

（3）给全局添加属性： 浏览器的全局对象是window，Node的全局对象是global。var声明的变量为全局变量，并且会将该变量添加为全局对象的属性，但是let和const不会。

（4）重复声明： var声明变量时，可以重复声明变量，后声明的同名变量会覆盖之前声明的遍历。const和let不允许重复声明变量。

（5）暂时性死区： 在使用let、const命令声明变量之前，该变量都是不可用的。这在语法上，称为暂时性死区。使用var声明的变量不存在暂时性死区。（6）初始值设置： 在变量声明时，var 和 let 可以不用设置初始值。而const声明变量必须设置初始值。

（7）指针指向： let和const都是ES6新增的用于创建变量的语法。 let创建的变量是可以更改指针指向（可以重新赋值）。但const声明的变量是不允许改变指针的指向。

(记得什么答什么简要概括)

42.Js的数据类型

基础数据类型（Undefined、Null、Boolean、Number、String）

引用数据类型（Object）

43.==和===的区别

当进行双等号比较时候： 先检查两个操作数数据类型，如果相同， 则进行===比较， 如果不同， 则愿意为你进行一次类型转换， 转换成相同类型后再进行比较， 而===比较时， 如果类型不同，直接就是false.(理解简单记)

双等号==：

　　（1）如果两个值类型相同，再进行三个等号(===)的比较

　　（2）如果两个值类型不同，也有可能相等，需根据以下规则进行类型转换在比较：

　　　　1）如果一个是null，一个是undefined，那么相等

　　　　2）如果一个是字符串，一个是数值，把字符串转换成数值之后再进行比较

　　三等号===:

　　（1）如果类型不同，就一定不相等

　　（2）如果两个都是数值，并且是同一个值，那么相等；如果其中至少一个是NaN，那么不相等。（判断一个值是否是NaN，只能使用isNaN( ) 来判断）

　　（3）如果两个都是字符串，每个位置的字符都一样，那么相等，否则不相等。

　　（4）如果两个值都是true，或是false，那么相等

　　（5）如果两个值都引用同一个对象或是函数，那么相等，否则不相等

　　（6）如果两个值都是null，或是undefined，那么相等

44.Ajax原理

使用XMLHttpRequest对象与Web服务器进行一步数据通信

使用JavaScript绑定和处理所有数据...

45.https跟http区别

传输信息安全性不同、连接方式不同、端口不同、证书申请方式不同

46.造成内存泄露的都有什么

就是没有使用，或已经使用完的变量，没有及时回收。

常见的javascript内存泄漏有:意外的全局变量,计时器或回调函数,dom清空或删除时，事件未清除导致的内存泄漏,闭包,子元素存在引起的内存泄漏

47.Js的垃圾回收机制

Javascript 具有自动垃圾回收机制，会定期对那些不再使用的变量、对象所占用的内存进行释放，原理就是找到不再使用的变量，然后释放掉其占用的内存。

JavaScript中存在两种变量：局部变量和全局变量。全局变量的生命周期会持续要页面卸载；而局部变量声明在函数中，它的生命周期从函数执行开始，直到函数执行结束，在这个过程中，局部变量会在堆或栈中存储它们的值，当函数执行结束后，这些局部变量不再被使用，它们所占有的空间就会被释放。不过，当局部变量被外部函数使用时，其中一种情况就是闭包，在函数执行结束后，函数外部的变量依然指向函数内部的局部变量，此时局部变量依然在被使用，所以不会回收

浏览器通常使用的垃圾回收方法有两种：标记清除，引用计数。

48.用户单点登录

用户登录时，验证用户的账户和密码

生成一个Token保存在数据库中，将Token写到Cookie中

将用户数据保存在Session中

请求时都会带上Cookie，检查有没有登录，如果已经登录则放行

49.大文件上传怎么上传

<https://www.cnblogs.com/cangqinglang/p/12204478.html>

就是大文件切成小文件,进行断点上传

核心是利用 Blob.prototype.slice 方法，和数组的 slice 方法相似

50.http状态码

200（OK） - 表示已在响应中发出

204（无内容） - 资源有空表示

301（Moved Permanently） - 资源的URI已被更新

303（See Other） - 其他（如，负载均衡）

304（not modified）- 资源未更改（缓存）

400 （bad request）- 指代坏请求（如，参数错误）

404 （not found）- 资源不存在

406 （not acceptable）- 服务端不支持所需表示

500 （internal server error）- 通用错误响应

503 （Service Unavailable）- 服务端当前无法处理请求

(记得多少说多少 )

51.Get和post的区别

GET在浏览器回退时是无害的，而POST会再次提交请求。

GET产生的URL地址可以被Bookmark，而POST不可以。

GET请求会被浏览器主动cache，而POST不会，除非手动设置。

GET请求只能进行url编码，而POST支持多种编码方式。

GET请求参数会被完整保留在浏览器历史记录里，而POST中的参数不会被保留。

GET请求在URL中传送的参数是有长度限制的，而POST么有。

对参数的数据类型，GET只接受ASCII字符，而POST没有限制。

GET比POST更不安全，因为参数直接暴露在URL上，所以不能用来传递敏感信息。

GET参数通过URL传递，POST放在Request body中。(根据自己理解答几点就行)

52.http的请求头有什么

Accept:浏览器能够处理的内容类型

Accept-Charset:浏览器能够显示的字符集

Accept-Encoding：浏览器能够处理的压缩编码

Accept-Language：浏览器当前设置的语言

Connection：浏览器与服务器之间连接的类型

Cookie：当前页面设置的任何Cookie

Host：发出请求的页面所在的域

Referer：发出请求的页面的URL

User-Agent：浏览器的用户代理字符串

(记住常用的即可)