**【 常见集锦】**

**ES6的语法**

1. 箭头函数
2. 解构赋值
3. 函数参数默认值
4. 模板字符串
5. let
6. Const
7. promise

**this的指向**

1. 作为函数调用，非严格模式下，this指向window，严格模式下，this指向undefined；

2. 作为某对象的方法调用，this通常指向调用的对象。

3. 使用apply、call、bind 可以改变绑定this的指向。

4. 在构造函数中，this指向新创建的对象

5. 箭头函数没有单独的this值，this在箭头函数创建时确定，它与声明所在的上下文相同。

**js的数据类型**

1.基本类型：string、number、boolean、undefined、null

2.引用类型：object、function、array

**作用域**

全局作用域：最外层函数定义的变量拥有全局作用域 指向window

函数作用域：专门保存仅函数内可用的局部变量的区域

**作用域链**

1.函数对象中保存函数调用时所有可用的作用域对象的链式结构

一个函数可用的所有变量（局部变量和全局变量）

作用域链控制着: 变量的使用顺序（先局部后全局！）

**原型链 原型对象**

由多级父对象，逐级继承，形成的链式结果，原型链保存着一个对象可用的所有属性和方法

每个被实例对象都有\_\_proto\_\_对象，它指向了构造该对象的构造函数的prototype属性。同时该对象可以通过\_\_proto\_\_对象来寻找不属于自身的属性

原型链控制着属性和方法的使用顺序：先自有，再共有——就近原则

替所有子对象保存共有方法和属性值的父对象

原型对象中的this指谁？

判断this不要看定义在哪儿，应该看何时被谁调用！

**js变量函数声明提前**

1.在程序正式执行之前，会将**var声明的变量(并未赋初始值)和function声明的函数**预读到当前作用域的顶部。

**如何水平垂直居中**

1.固定宽高：使用定位 + margin

2.相对大小：transform: translate(-50%, -50%) 定位 （有兼容性）

3.弹性：主轴和副轴上设置居中

4.vertical-align：middle

**冒泡排序**

*var* arr = [7, 6, 5, 4, 3, 2, 1];

        for (*var* i = 0; i < arr.length - 1; i++) {

            //2.指定轮数和次数（内循环控制次数）

            for (*var* j = 0; j < arr.length - 1; j++) {

                //3.判断是否符合标准。如果符合标准交换位置。

                //从小到大排列顺滑，如果前面的比后面的大，那么交换位置。

                if (arr[j] > arr[j + 1]) {

*var* temp = arr[j];

                    arr[j] = arr[j + 1];

                    arr[j + 1] = temp;

                }

            }

        }

        console.log(arr);

**递归**

*var* count=0;

*function* say(){

  console.log('然哥在讲故事，故事的内容是');

  //每讲1次加1

  count++;

  //如果讲到第3次,也就是count为3的时候，结束函数

  if(count===3){

    return;//阻止往后执行

  }

  //递归：在函数内调用自身

  say();

}

say();

**数组去重**

1.for循环嵌套，利用splice去重

2.建立新数组，用indexOf去重

3.ES6用set去重

*var* arr=[1,1,2,2,3,3];

*function* newArr(*arr*){

            for(*var* i=0; i<*arr*.length; i++){

                for(*var* j=i+1; j<*arr*.length; j++){

                    if(*arr*[i]==*arr*[j]){

*arr*.splice(j,1)

                    }

                }

            }

            return *arr*;

        }

        console.log(newArr(arr));

**js迭代的方法 或数组循环的方法**

1.every（）：对数组中的每一项运行给定函数，如果该函数对每一项都返回true,则返回true。

2.filter（）：对数组中的每一项运行给定函数，返回该函数会返回true的项组成的数组。

3.forEach（）:对数组中的每一项运行给定函数。这个方法没有返回值。

4.map（）：对数组中的每一项运行给定函数，返回每次函数调用的结果组成的数组。

5.some（）：对数组中的每一项运行给定函数，如果该函数对任一项返回true，则返回true。

**ES5和ES6的区别**

1.let声明变量和const声明常量，两个都有块级作用域ES5中是没有块级作用域的，并且var有变量提升，在let中，使用的变量一定要进行声明

2.箭头函数ES6中的函数定义不再使用关键字function()，而是利用了()=>来进行定义

3.模板字符串模板字符串是增强版的字符串，用反引号（`）标识，可以当作普通字符串使用，也可以用来定义多行字符串

**外边距重叠**

两个或多个块级盒子的垂直相邻边界会重合

1. 全部都为正值，取最大者；  
   b、不全是正值，则都取绝对值，然后用正值减去最大值；  
   c、都是负值，则都取绝对值，然后用0减去最大值。

解决：

1. 外层元素padding代替
2. 内层元素透明边框 border:1px solid transparent;

**怎样给数组添加或删除新元素**

**添加**

1.push()可把一个参数顺序添加到 数组对象的的尾部，它直接修改了 数组对象，而不是创建一个新的数组，返回新的长度

2.unshift()可把一个参数顺序添加到 数组对象的头部

**删除**

1. splice()删除元素

**setTimeout和setInterval的区别**

setTimeout在规定时间后执行完某个操作就停止了，而setInterval则可以一直循环下去

**什么是闭包，对闭包的了解**

1.外层函数调用后，外层函数的作用域对象被内层函数引用着，无法释放，就形成了闭包

闭包的优点：重用率高，防止全局污染缺点：但是占用内存大，需要尽快释放：内层函数=null

**跨域，****如何解决跨域请求**

**跨域**

浏览器不能执行其他网站的脚本。它是由浏览器的同源策略造成的，是浏览器施加的安全限制。

**如何解决跨域请求**

1. jsonp jsonp只支持get请求

2.引入 cors 模块 const cors = require('cors');

3.代理

**对Promise的理解**

**Promise**

1.解决回调地狱问题，保证多个异步任何都可以顺序执行的技术

何时：只要多个异步任务按顺序执行时，都可用promise执行

优势：即可按顺序执行，又不会产生嵌套

### 缺点：

### 1.一旦新建就会立即执行，无法中途取消

2.如果不设置回调函数，promise内部的错误无法反映到外部

**vuex中commit和dispatch的区别**

1.dispatch含有异步操作，例如向后台提交数据，写法： this.$store.dispatch('action方法名',值);

2.commit 同步操作，写法：this.$store.commit('mutations方法名',值)

**vue组件间的传参 父传子 子传父 兄弟之间**

**父传子** props：子定义用props定义公开的属性用于接住父给的变量 父级需要用绑定语法将自己的变量绑定给子组件定义的公开属性

**子传父** $emit ：父组件需要在引用的子组件上添加一个自定义事件，并绑定父组件的一个处理函数； 子组件需要通过$emit（“父级的自定义事件名”,参数值） 触发父级的自定义事件this.$emit(“自定义事件”)

**兄弟之间**

1. 新建一个js文件作为兄弟组件间传值的中转站 组件1把值先传到bus 组件2从bus接收值

**移动端页面适应方案**

1. 媒体查询
2. flexible.js
3. 视口宽度

**vue的绑定原理**

访问器属性 + 虚拟DOM树

**vue的双向绑定原理**

1.vue数据双向绑定是通过数据劫持结合发布者-订阅者模式的方式来实现的，数据和视图同步更新

2.核心：Object.defineProperty()

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，触发相应监听回调。

**对项目进行估哪些优化**

1.懒加载

2.v-if v-show区分使用场景

3.v-for加key和不加key的区别

**封装过哪些组件**

1.音乐播放进度条

2.轮播图

**什么是单页面应用**

整个页面只有一个完整的HTML，其他所谓的页面，只是一个页面中的片段或者是组件；页面跳转，其实就是在一个HTML页面中切换不同的片段内容或者是组件

**如何实现单页面应用**

1.创建一个完整的页面

2.分别为每个页面定义一个子组件对象

3.路由器监视地址栏的变化,根据地址栏变化找到对应的组件

**单页面应用---页面传递数据**

1.修改路由字典数组中的路由对象定义

{path:”/相对路径/:变量名”，component:页面组件对象名，props:true}

props:true 让变量值自动成为下个页面props中的自定义属性成为参数在下个页面中用于绑定

1. 用props[“变量名”]来接住
2. 页面获取参数：this.$route.params.id
3. HTML跳转时<router-link to=”/相对路径/变量值”> 变量值不要加冒号：
4. js跳转 this.$router.push(”/相对路径/变量值”) 变量值不要加冒号：

**单页面应用的优缺点**

单页面的优点：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；

前后端分离；页面效果会比较炫酷（比如切换页面内容时的专场动画）。

单页面缺点：首屏加载慢，初次加载时耗时多；

解决：异步延迟下载，（打包时，不要把所有组件打包在一个大的文件，应该每个组件分开打包为单独文件； 首屏加载时只强制加载第一个组件的内容，后续程序由异步延迟下载）

懒加载（打包时，不要把所有组件打包在一个大的文件，应该每个组件分开打包为单独文件 首屏加载时，不会下载其他组件，只有切换到才下载）

**vue相关的组件之间的通信方式**

1.父组件传给子组件

2.子组件传给父组件

3.建立新的VUE对象

4.使用vuex

5.使用全局函数

**回流和重绘，**

1.当DOM的变化影响了元素的几何属性，浏览器会使渲染树中受到影响的部分失效，并重新构造渲染树。这个过程称为重排,完成重排后，浏览器会重新绘制受影响的部分到屏幕，该过程称为重绘。

**防抖和节流**

比如轮播图点太快了，浏览器会反应不过来，设一个定时器来限制轮播反映的时间

**观察者模式**

当一个变量被被修改时，可以自动通知所有关注这个变量的其他对象，让他们自动重新获得这个变量的新值

**keep-alive的两种函数的作用**

vue的一个内置组件，缓存组件内部状态，避免重新渲染

1.**include** - 字符串或正则表达式。只有匹配的组件**会被缓存**。  
2.**exclude** - 字符串或正则表达式。任何匹配的组件都**不会被缓存**。

**路由守卫的用法**

## 给路由添加一个meta meta：{auth: true} 用于判断是否需要 在全局守卫beforeEach判断要访问的组件是否需要登录 判断我们保存用户信息的位置 如果有值就放行 没有值就跳转到登录页

## **路由守卫的调用顺序：**

beforeRouteLeave (离开组件的)

beforeEach(全局路由)

beforeRouteEnter(跳转到的路由组件内的)

afterEach(全局路由)

vue实例的生命周期

**get和post的区别**

1.最直观的区别就是GET把参数包含在URL中，POST通过request body传递参数。

2.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。

3.GET请求只能进行url编码，而POST支持多种编码方式。

4.GET请求在URL中传送的参数是有长度限制的，而POST么有。

**vue生命周期**

创建，挂载，更新，销毁

**生命周期钩子函数**

创建，挂载，更新，销毁

1. beforeCreate创建前 可以操作DOM上的变量
2. created完成数据检测，属性和方法的运算
3. beforeMount
4. Mounted
5. beforeUpdate
6. Updated
7. beforeDestroy
8. Destroyed

**vuex**

vuex是vue中状态管理 相当于一个共享中心 其他组建中共有的数据放在里面公用

场景应用：单页面应用，组件之间的状态，音乐播放，登录状态的等

vuex的属性：

1.state(存放变量)

2.getter(从基本数据中派生出来的数据)

3.mutation(专门修改state中的变量)

4.action(专门负责发送ajax请求，从服务器端获取数据，异步操作)

5.Module(模块化vuex)

**sessionStorage 和localStorage 和cookie的区别**

1.sessionStorage:客户端技术可以在浏览器窗口存储数据,一但关闭窗口,数据就没了,如果多个窗口,数据无法共享;

2.localAtorage:客户端技术可以在浏览器窗口存储数据,数据操作是永久的

3.cookie:客户端技术可以在浏览器窗口存储数据,特点是有效性

1）相同点是都是保存在浏览器端、且同源的  
2）cookie数据始终在同源的http请求中携带（即使不需要），即cookie在浏览器和服务器间来回传递，而sessionStorage和localStorage不会自动把数据发送给服务器，仅在本地保存。cookie数据还有路径（path）的概念，可以限制cookie只属于某个路径下

**面向对象：封装，继承，多态。原理和应用**

封装：创建一个对象集中保存一个事物的属性和功能，便于大量数据的管理和维护，今后，只要使用面向对象方式编程，都要先创建对象。

继承：父对象中的成员，子对象无需重复创建，就可直接使用！同一类型的所有子对象，都需要相同的方法定义或属性值时，都会用继承的方式来实现。

多态：重载(overload): 一个函数，根据传入的实参值不同，执行不同的逻辑

重写(override)：在子对象中定义一个和父对象中成员同名的成员！ 因为从爹继承来的东西，不总是好用的！

**MVVM的理解**

vue是实现了双向数据绑定的mvvm框架，当视图改变更新模型层，当模型层改变更新视图层。在vue中，使用了双向绑定技术，就是View的变化能实时让Model发生变化，而Model的变化也能实时更新到View。

Model-View-ViewModel 的缩写

1.模型对象(Model): 专门保存界面所需的所有变量和事件处理函数的对象

2.界面(View): 增强版的HTML+CSS

3.视图模型(ViewModel): 一种专门自动将模型对象中的数据和函数与界面中保持同步的特殊的程序

ViewModel 监听模型数据的改变和控制视图行为、处理用户交互，简单理解就是一个同步View 和 Model的对象，连接Model和View。

**vue与Angular react的区别**

1.与AngularJS的区别

相同点：

都支持指令：内置指令和自定义指令；都支持过滤器：内置过滤器和自定义过滤器；都支持双向数据绑定；都不支持低端浏览器。

不同点：

AngularJS的学习成本高，比如增加了Dependency Injection特性，而Vue.js本身提供的API都比较简单、直观；在性能上，AngularJS依赖对数据做脏检查，所以Watcher越多越慢；Vue.js使用基于依赖追踪的观察并且使用异步队列更新，所有的数据都是独立触发的。

2.与React的区别

相同点：

React采用特殊的JSX语法，Vue.js在组件开发中也推崇编写.vue特殊文件格式，对文件内容都有一些约定，两者都需要编译后使用；中心思想相同：一切都是组件，组件实例之间可以嵌套；都提供合理的钩子函数，可以让开发者定制化地去处理需求；都不内置列数AJAX，Route等功能到核心包，而是以插件的方式加载；在组件开发中都支持mixins的特性。

不同点：

React采用的Virtual DOM会对渲染出来的结果做脏检查；Vue.js在模板中提供了指令，过滤器等，可以非常方便，快捷地操作Virtual DOM。

**H5新特性**

1.语义化标签 2.增强型表单 3.视频和音频 4.canvas绘图 5.svg矢量图 6.地图API 7.drag拖放 8.webworker，是一种运行在后台的javascript脚本，不会影响页面性能 9.webstorage（sessionStorage会话存储，localStorage持久性存储）10.websocket

**C3新特性**

1.渐变 2.过渡 3.媒体查询 4.2D 3D转换 5.弹性布局

**深拷贝和浅拷贝**

浅拷贝：修改了一个数组原数组也会改变 ，因为引用了同一个地址的数据

深拷贝：拷贝对象的具体内容，地址不一样，互不干涉

实现深拷贝的方法 :

1. JSON内置的方法：该方法是用JSON.parse将对象转为字符串，然后在用JSON.stringify转回对象json字符串转换为对象的时候，会自己去构建新的内存地址存放数据
2. Object的内置方法assign：对对象进行拼接
3. 递归实现

**dom优化**

1.减少重排重绘

2.添加顺序：尽量在appendChild前添加操作

3.合并dom操作：利用cssText ，就是把改变的样式写在一起

4.缓存布局信息

5.文档片段

**axios如何同步执行**

**async: false, // fasle表示同步请求，true表示异步请求**

**BFC是什么**

1.是一块区域body 根元素

2.浮动元素：float 除 none 以外的值

3.绝对定位元素：position (absolute、fixed)

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

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

**jquery的特点及好处**

1.链式操作 自带for循环 兼容大部分的浏览器

**css的引入方式**

1.行内样式、内部样式表、外部样式表。

**说出至少4种vue当中的指令和它的用法？**

1.v-once: 只绑定一次

2.v-bind: 绑定数据

3.v-model: 绑定模型

4.v-on: 绑定事件

5.v-if v-show: 条件渲染

**webstore的优点**

1.没有同源策略的限制，客户端可以与任何服务器之间进行通讯, 数据格式轻量，通讯高效

WebSocket协议的前缀是ws(如果加密的话，则为wss)

**token**

1.令牌，最大的特点就是随机性，不可预测。

用处：

1. 防止表单重复提交
2. anti csrf攻击（跨站点请求伪造）
3. 一般的做法是请求接口的时候，获取最新的Token，然后在发送请求的时候，把Token传过去。一般是在全局的$ajax或者axios中封装
4. 抵挡csrf攻击的方法：token 验证码 refer头

**虚拟DOM**

优点：

1. 体积小 2.遍历查找快 3.修改效率高 4.避免重复编码

**vue更新数组时触发视图更新的方法**

答:push()；pop()；shift()；unshift()；splice()；sort()；reverse()

**怎样定义vue-router动态路由**

this.$route.params.id

**第一次页面加载会触发哪几个钩子？**

答：beforeCreate， created， beforeMount， mounted

**[如何实现路由拦截](https://blog.csdn.net/weixin_30508241/article/details/99923904?biz_id=102&utm_term=vue%E8%B7%AF%E7%94%B1%E6%8B%A6%E6%88%AA%E5%A6%82%E4%BD%95%E5%AE%9E%E7%8E%B0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduweb~default-0-99923904&spm=101)**

定义路由的时候就需要多添加一个自定义字段meta meta:{requireAuth:true} 用于判断该路由的访问是否需要登录（或者存在）如果用户已经登录，则顺利进入路由， 否则就进入登录页面。然后再main.js 判断该路由是否需要登录权限 判断本地是否存在access\_token

**如何axios实现请求和响应拦截**

请求拦截器：this.$axios.interceptors.request.use

响应拦截器：this.$axios.interceptors.response.use

**computed和watch methods的区别**

****与computed的区别是，watch更加适用于监听某一个值的变化并做对应的操作，比如请求后台接口等，而computed适用于计算已有的值并返回结果****

Computed: **computed计算属性具有缓存功能** 数据量大，需要缓存的时候用computed

Watch: watch侦听器是侦听一个特定的值，**当该值改变时执行特定的函数**执行 异步操作，开销比较大，避免堵塞主线程

Methods: methods是方法 只要重新渲染，都会执行调用的methods每次确实需要重新加载，不需要缓存时用methods。

**js中如何将object转化为json数据，json数据转js对象**

**js对象转json数据:  JSON.stringify();**

**json数据转js对象:  JSON.parse();**

**浏览器渲染的过程**

1. **DNS查询**
2. **TCP链接**
3. **HTTP响应**
4. **服务器响应**
5. **客户端渲染**

**浏览器渲染的过程**

DNS解析

TCP连接

发送HTTP请求

服务器处理请求并返回HTTP报文

浏览器解析渲染页面

连接结束

**Axios请求拦截和响应拦截的应用场景**

请求拦截器：当我们在做接口请求时，比如判断登录超时时候，通常是接口返回一个特定的错误码，那如果我们每个接口都去判断一个耗时耗力，这个时候我们可以用拦截器去进行统一的http请求拦截。

响应拦截器：响应拦截器的作用是在接收到响应后进行一些操作，例如在服务器返回登录状态失效，需要重新登录的时候，跳转到登录页。